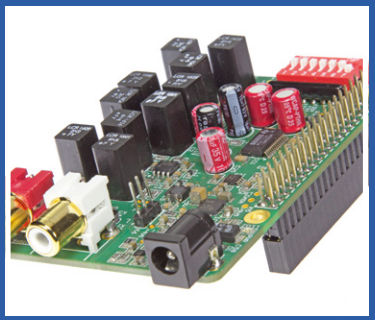
Juli/August 2017 | Nr. 556
www.elektormagazine.de

elektor

LEARN DESIGN SHARE



Audio-DAC für Raspberry Pi

Netzwerk-Audiospieler mit Volumio

Wirklich gedruckte Schaltungen

Platinendrucker/Reflow-Ofen Voltera V-One

FPGA/DSP-Board für Schmalband-SDR

Ein voll-programmierbares Empfänger/Sender-Basisboard

nWatch

ein Wearable-Entwicklungssystem

Oder: Wie ich meine eigene SmartWatch baute

In dieser Ausgabe:
6 Labor-Projekte
9 Homelab-Projekte
3x Grundlagen
und vieles mehr!

Die Kunst des Routens: Tipps für das Platinen-Layout • Der große Bruder des ESP8266: Erste Schritte mit dem ESP32 • **Zwei-Draht-Dimmer** • **Elektronik im All** • BBC micro:bit für Elektroniker: Datenlogger und Oszilloskop • **Röhren-Kennlinienschreiber** • **MicroPython und das Pyboard** • LoRaWAN: Architektur, Protokoll, Sicherheit und Betrieb • **Audioverstärker Klasse A** • Ein fast universeller IR-Sender • **Schaltbarer HF-Abschwächer** • **Positionslichter für Flugmodelle** • Sniffer mit dem Radio-Modul RFM12 • **5 V aus erschöpften Alkali-Batterien** • Meeresrauschen-Simulator

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

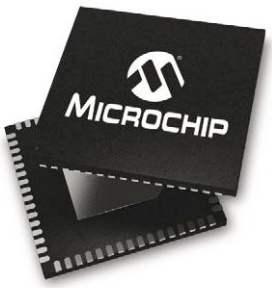
4 198630 314904 08

Komplexe Designs durch PIC® MCUs mit integrierter Peripherie vereinfachen

PIC18F „K42“ MCUs bieten umfangreichste Core-unabhängige Peripherie und größten Gesamtspeicher

Mit dem höchsten Anteil an Core-unabhängiger Peripherie (CIPs) sorgen die MCUs der PIC18F-„K42“-Serie für mehr Systemleistung bei reduziertem Code – selbst in komplexen Designs.

Die Systemleistung wird durch Direct Memory Access und vektorisierte Interrupts mit fester Latenz verbessert, wobei trotz reduziertem Code eine schnelle Datenverarbeitung garantiert wird. Zusätzlich bieten diese MCUs den größten Gesamtspeicher aller 8-Bit PIC® MCUs und das zusammen mit präzisen Analogschaltungen, flexiblen Stromsparmodi und serieller Kommunikation.



microchip DIRECT
www.microchipdirect.com

MICROCHIP

www.microchip.com/EUPIC18FK40

Der Name Microchip, das Logo und PIC sind eingetragene Warenzeichen der Microchip Technology Incorporated in den USA und in anderen Ländern. MiWi, mTouch und ZENA sind Marken der Microchip Technology Incorporated in den USA und in anderen Ländern. Alle anderen hier erwähnten Marken sind im Besitz der jeweiligen Eigentümer. © 2017 Microchip Technology Inc. Alle Rechte vorbehalten. DS30010138A. MEC2125Ger06/17

Impressum

48. Jahrgang, Nr. 556
Juli/August 2017

Erscheinungsweise: 12 x jährlich
(Elektor Doppelheft im Wechsel mit dem Elektor Business Magazin)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 02 41/955 09 190
Fax 02 41/955 09 013

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Hauptsitz des Verlags
Elektor International Media
Allee 1, NL-6141 AV Limbricht

Anzeigen:
Margriet Debeij (verantwortlich)
Tel. 02 41/955 09 174 / Fax 02 41/955 09 013
Mobil: +49 (170) 550 539 6
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/955 09 177 / Fax 02 41/955 09 013
Mobil: +31 6 510 530 39
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste ab 01.01.2017.

Distribution:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0
Fax 0 22 25/88 01-199

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für alle ausländischen Ausgaben inklusive Lizenzen. Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© 2017 elektor international media b.v.
Druck: Senefelder Misset, Doetinchem (NL)
ISSN 0932-5468

Begeisternde Projekte

Auf der letzten *electronica* hatten Start-ups und innovative Maker aus der ganzen Welt Gelegenheit, ihre Prototypen und Ideen einem breiten Publikum vorzustellen und um den „Fast Forward Award“ zu wetteifern. Als Mitglied der Jury habe ich mir alle bei uns eingereichten Einsendungen schon vor der Messe angesehen und eine Bewertung abgegeben, die in die Endpunktzahl der Teilnehmer eingeflossen ist. Ich kann mich erinnern, dass ich von einigen Projekten so begeistert war, dass ich einfach die Höchstzahl von fünf Punkten vergeben musste. Darunter war eine intelligente Armbanduhr, die ein junger Tüftler von Grund auf selbst konzipiert hatte. Die Entwicklungsschritte reichten von der Festlegung der Spezifikationen und der Auswahl der Bauteile über das knifflige Platinendesign und das Schreiben der Firmware bis zum Entwurf des 3D-gedruckten Gehäuses.

Ich freue mich daher besonders, das Projekt in dieser Ausgabe als Titelgeschichte vorstellen zu können. Sicherlich wird der Nachbau nur den besonders fortgeschrittenen Lesern möglich sein, die Eigeninitiative beim Besorgen der Bauteile und genug Geduld und Spucke für das kleinteilige Bestücken mitbringen. Doch ist dieser Artikel auch für alle anderen Elektroniker interessant, denn er zeigt, dass begeisterte Maker selbst auf derart schwierigem Terrain Erfolg haben können.

Und wir haben natürlich wie in jeder Ausgabe auch einfachere Projekte zu bieten. Der Schaltbare Abschwächer auf Seite 18 ist ein nützliches Werkzeug für alle Messtechniker und kommt mit einem Minimum an Bauteilen aus. Mit einer überschaubaren Stückliste ist unser Meeresrauschen-Simulator auf Seite 35 für alle Einsteiger geeignet – eine Platine gibt es im Elektor-Shop. Und schauen Sie sich doch gleich einmal auch den Audioverstärker auf Seite 112 und die Modellbau-Schaltung auf Seite 122 an!

Viel Spaß beim Lesen wünscht

Jens Nickel
Chefredakteur Elektor

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf
Leserservice:	Ralf Schmiedel
Korrekturen:	Malte Fischer
Internationale Redaktion:	Thijs Beckers, Eric Bogers, Jan Buiting, Mariline Thiebaut-Brodier
Elektor-Labor:	Ton Giesberts, Hedwig Hennekens, Luc Lemmens, Jan Visser, Clemens Valens
Grafik & Layout:	Giel Dols

Inhalt

48. Jahrgang – Nr. 556
Juli/August 2017

Rubriken

3 **Impressum**

29 **Tipps und Tricks**
Von Lesern für Leser

59 **HomeLab Helikopter**

68 **Projekt 2.0**
Korrekturen, Updates und Leserbriefe

79 **Elektormagazine.de/Labs**
Projekte in der Pipeline

100 **Retronik**
Die Zweistrahl-Oszilloskope Tektronix 556 und 565

126 **Elektor Ethics**
Smarte Geräte!

130 **Zenerdoku**

Hintergrund

6 **Die Kunst des Routens**
Tipps für das Platinen-Layout

38 **Der I²C-Bus**
Teil 2: Der Bus am Controller

44 **Wirklich gedruckte Schaltungen**
Platinendrucker/Reflow-Ofen Voltera V-One

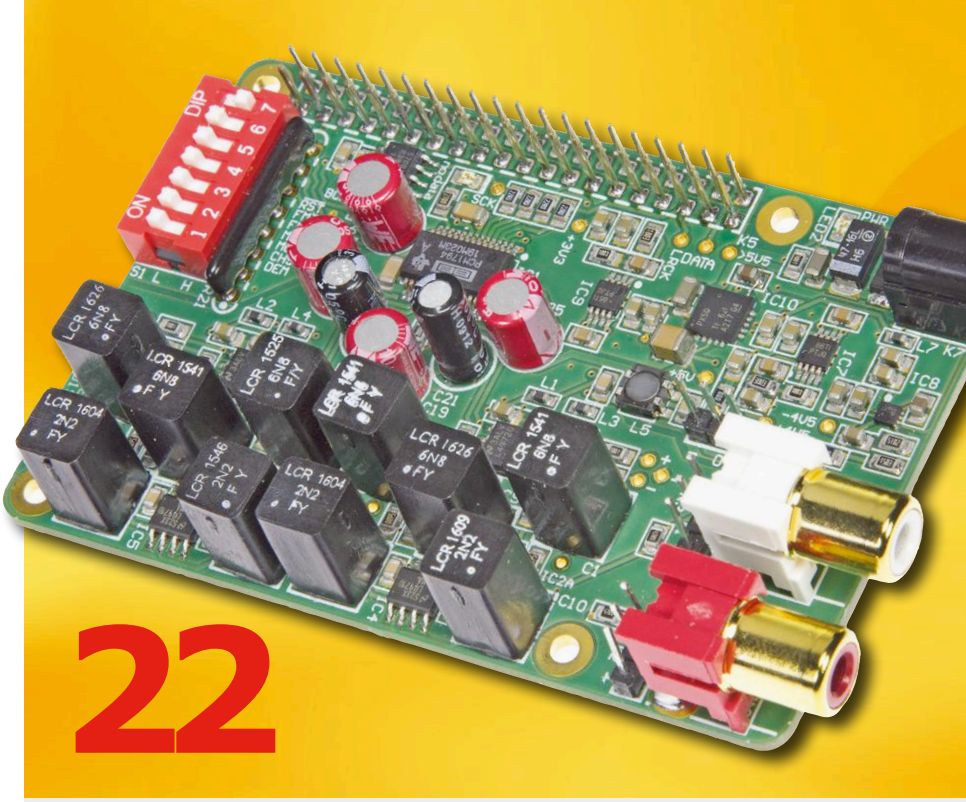
85 **Q&A**
Bauteile in der Raumfahrt

88 **Der große Bruder des ESP8266**
Erste Schritte mit dem ESP32 und der Arduino-IDE

94 **Elektronik im All**
Weltraumforschung für jedermann

99 **Aus dem Labor**
Wo bitte ist das Menü?

107 **LoRaWAN**
Architektur, Protokoll, Sicherheit und Betrieb



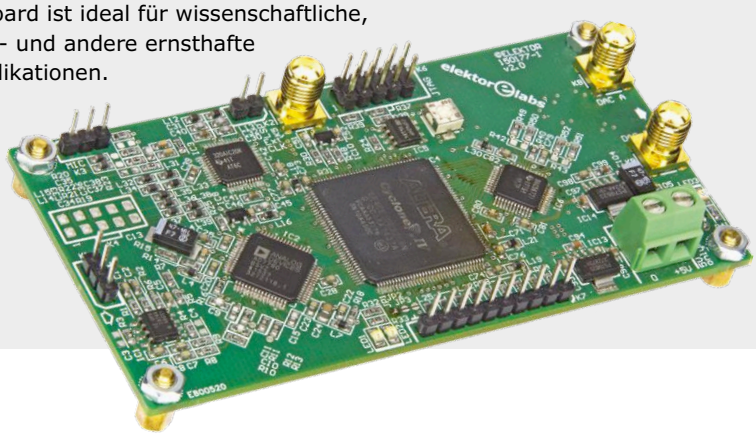
22

FPGA/DSP-Board für Schmalband-SDR

70

Ein voll-programmierbares Empfänger/Sender-Basisboard

Dieses DSP-Board auf FPGA-Basis enthält alles, um die Basisbandverarbeitung von SSB/CW- und AM-Funksignalen durchführen zu können. Es löst viele der Probleme und Einschränkungen einfacher SDR-Plattformen und bietet eine perfekte Grundlage für einen Transceiver von der Kurz- bis zur Mikrowelle. Unser Board ist ideal für wissenschaftliche, Amateur- und andere ernsthafte Funkapplikationen.



Projekte

14 **Zwei-Draht-Dimmer**
Steuert alles: LEDs, Glühlampen und Heizlüfter

18 **Schaltbarer HF-Abschwächer**
Definierte Pegel durch einstellbare Abschwächung

22 **Audio-DAC für RPi**
Netzwerk-Audiospieler mit Volumio

44

Audio-DAC für Raspberry Pi

Netzwerk-Audiospieler mit Volumio

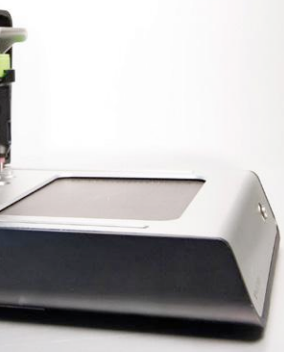
Der große Bruder des ESP8266

88

Erste Schritte mit dem ESP32 und der Arduino-IDE

Der ESP8266 von Espressif beeindruckte die Entwicklergemeinde mit seinen geringen Kosten: Bei AliExpress konnte man das Bauteil schon um zwei oder drei Euro erhalten. Mit dem großen Bruder ESP32 möchte Espressif seine WLAN-SoCs für die Zukunft fitmachen: Neben einem Zweikernprozessor gibt es nun ein Bluetoothmodul, mehr Arbeitsspeicher und Erweiterungen im Bereich IO. In diesem Artikel stellen wir die Programmierung des ESP32 mit der Arduino-IDE vor. Eine Besprechung des nativen Entwicklungstools IDF folgt im nächsten Heft.

Platinendrucker/ Reflow-Ofen Voltera V-One



30 BBC micro:bit für Elektroniker
Datenlogger und Oszilloskop

35 Meeresrauschen-Simulator
Sanft einschlafen im Rauschen der Wellen

48 nWatch, ein Wearable-Entwicklungssystem
Oder: Wie ich meine eigene SmartWatch baute

62 Fünf Volt aus erschöpften Alkali-Batterien
Den letzten Tropfen „Saft“ rausquetschen

63 Röhren-Kennlinienschreiber
Remake eines Tektronix 570

70 FPGA/DSP-Board für Schmalband-SDR
Ein voll-programmierbares Empfänger/ Sender-Basisboard

80 MicroPython und das Pyboard
Von der blinkenden LED zur ... blinkenden LED am Webserver

104 Sniffer mit dem Radio-Modul RFM12
Belauschen des Funkverkehrs

112 Audioverstärker Klasse A
Unkonventionell mit Spannungsreglern LM317

114 Ein fast universeller IR-Sender

118 Ukule-LED
Ukulele lernen mit seriell adressierbaren RGB-LEDs

122 Vielsagende Positionslichter
Für alle Flugmodelle

elektor magazine

Vorschau

Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- Funktionsgeneratoren
- Datenlogger für Heizung
- Küchenwaage mit Arduino
- Funkplatine für das FPGA/DSP-Board
- Flexibles IoT-Interface
- Raspberry Pi Zero W in der Praxis
- Peltier-Lampe
- iPhone-Programmierung für Elektroniker

Und vieles mehr!

Änderungen vorbehalten.
Elektor September/Oktober 2017 erscheint am 16. August 2017.
Verkaufsstellen findet man unter www.pressekaufen.de.

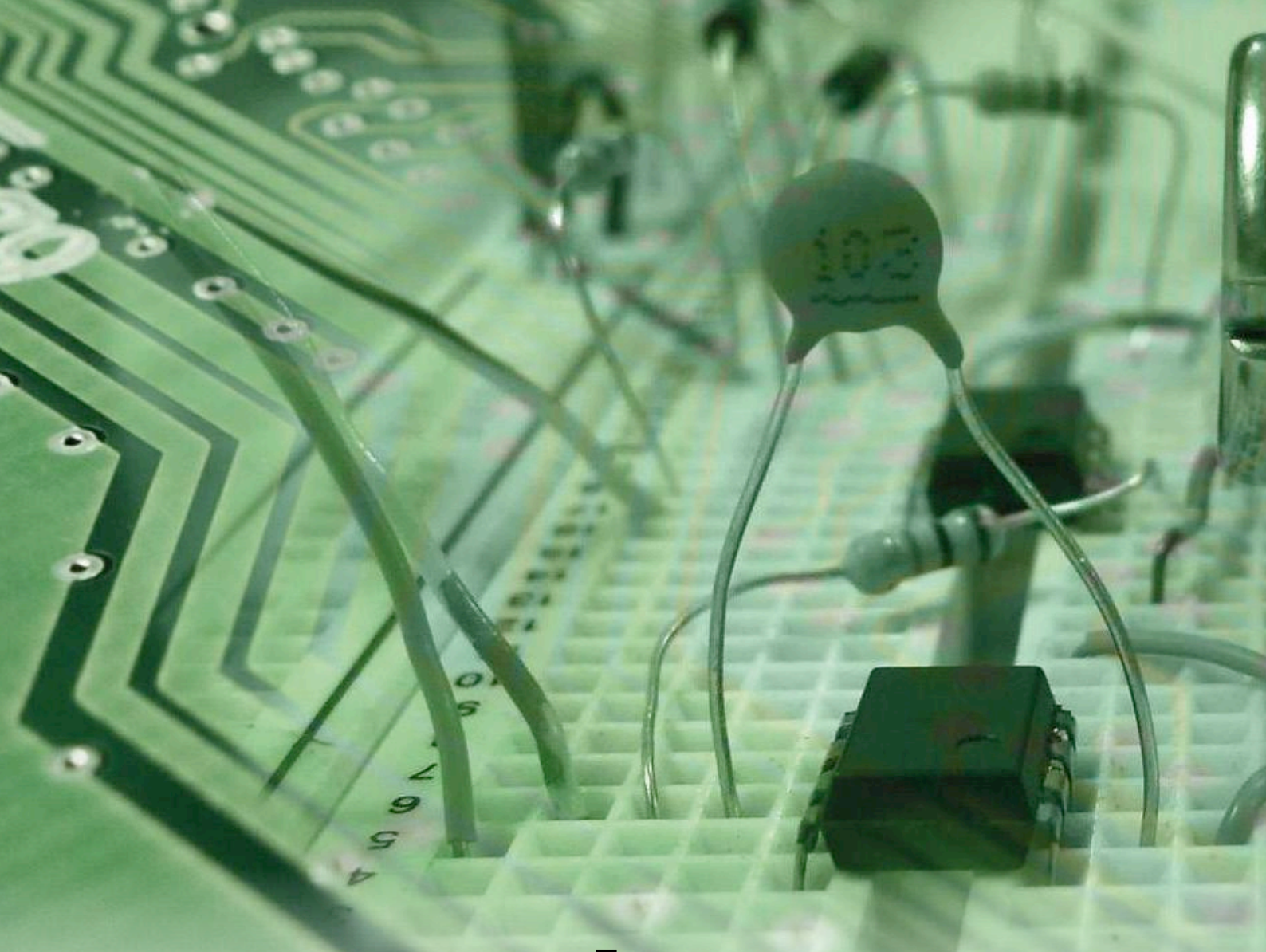
Vorschau Elektor Business

LEDs & Displays

- Weiße und farbige High-Power-LEDs
- Kompakter 48-W-Treiber
- LED-Scheinwerfer für Rinspeed Concept Car

Und vieles mehr!

Änderungen vorbehalten.
Elektor Business 4/2017 erscheint am 15. Juli 2017.



Die Kunst des Routens

Tipps für das Platinen-Layout

Von **Clemens Valens** (Elektor-Labor)

Auch einfache Schaltungen haben gute Platinen verdient. Eine gute Platine garantiert nicht nur Zuverlässigkeit und dass sich die Schaltung erwartungsgemäß verhält, sie reduziert auch Störstrahlung und stellt für Test und Reparatur Testpunkte zur Verfügung. Außerdem passt so die Elektronik gut in das vorgesehene Gerät. Eine Platine ist nämlich nicht nur eine Bauteileunterlage, sondern ein elementarer Teil des Systems. Das Layout muss nicht unbedingt dem Entwickler gefallen, sondern dem Kunden und Endanwender dienlich sein, wer immer das auch ist (**Bild 1**).

Der im folgenden verwendete Jargon und die Akronyme sind im **Glossar** zu diesem Artikel erläutert. Und auch wenn dieser Artikel umfangreich ist, so ist er doch alles andere als erschöpfend. Platinen-Layout ist eigentlich ein zu komplexes Thema, um es komprimiert in einem einzigen Artikel aufbereiten zu können.

Systemintegration

Der erste Schritt — besonders gerne ignoriert — besteht darin, die Integration ins System zu untersuchen. Welche Form muss die Platine haben? Wo sollen die Befestigungslöcher sein? Und wie viele? Wie sieht es mit Kabeln und Leitungen aus? Von wo kommen sie und wo führen sie hin? Gibt es Höhenbegrenzungen? Drohen thermische Probleme? Gibt es Interaktionen des Anwenders mit dem Board und wenn ja welche? Soll die Platine auf die Rückseite, nach vorne, den Boden oder an eine Seite? Braucht es Steckverbinder? Sitzen Drehencoder, Potis oder Taster auf der Platine? Ein Display? LEDs? Harren da Probleme mit dem Gehäusematerial? Und nicht zuletzt: Was ist mit dem Netzteil?

Selbst wenn ein System nur aus einer Platine in einem Gehäuse besteht, sollte man das Gehäuse auswählen, bevor man mit dem Layout der Platine beginnt. So kann man die Form und Größe der Platine rechtzeitig festlegen und muss sich nicht ärgern, wenn sie schon fertig ist.

Für viele ist das Layouten einer Platine eher ein Detail oder eine Nebensache. Die Schaltung selbst wurde zwar mit viel Aufwand getestet, doch dann wird das Resultat ohne viel Federlesens in eine Platine gegossen, weil sich so die Bauteile einfacher irgendwo befestigen lassen. Wir zeigen, wie Sie es besser machen!

Wer sich nicht wirklich gut mit der Verarbeitung von Metall, Holz und Kunststoff auskennt und auch keinen Zugang zu einer CNC- bzw. Laser-Schneidemaschine hat, sollte mechanische Arbeiten möglichst auf ein Minimum begrenzen.

Platinenherstellung

Eine weitere wichtige Frage: Wer stellt die Platine her? Wollen Sie die Platine selbst zuhause ätzen? Sind Sie dafür fit genug? Wenn nicht, vergessen Sie Durchkontaktierungen (Vias) und beschränken sich auf einseitige Platinen. Schon bei zweiseitigen Exemplaren müssen die Filme für Ober- und Unterseite perfekt ausgerichtet sein. Die Metallisierung von Vias ist ein aufwändiger Prozess, vor allem wenn es viele davon gibt. Trotzdem erzielen viele Selberätzer tolle Platinen – es hängt eben alles vom Können ab.

Es empfiehlt sich auch, genügend Abstand zu Masse oder sonstigen Kupferflächen zu halten, denn ohne Lötstoppmaske ist das nachträgliche Aufspüren und Entfernen von ungewollten Lötzinnbrücken kein Spaß. Es hilft deshalb, Leiterbahnen mit

Bild 1. Ihre Platine wird bei einer Auftragsbestückung vielleicht in solch einer weit entfernten Fabrik bestückt. Zur Vermeidung von Komplikationen sollte sie dann mit so wenig Erläuterung wie möglich produzierbar sein.

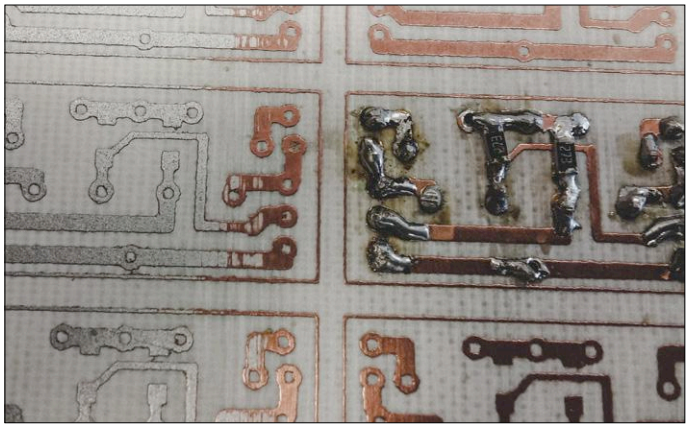
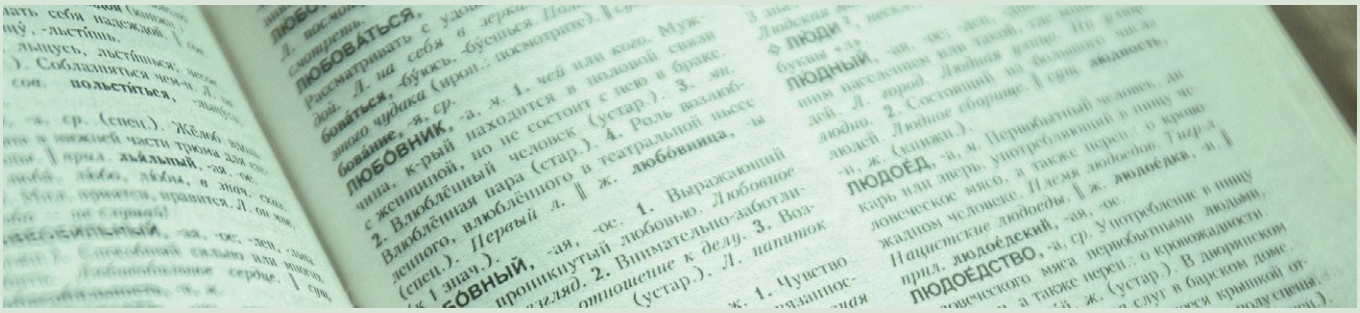


Bild 2. Eine selbstgeätzte Platine mit vielen noch zu bohrenden Löchern und zu erledigenden Sägearbeiten. Wollen Sie sich das wirklich antun?

genug Abstand zu verlegen. Und wenn man nicht sehr gut ausgerüstet ist, empfehlen sich größere Pads, weil so das Bohren einfacher ist (**Bild 2**).

Eine Platinen-Fräsmaschine stellt zwar ähnliche Anforderungen, doch wenn man nicht sehr erfahren im Ätzen ist, erreicht man damit eine höhere Präzision. Damit bekommt man zwar keine sehr hohe, aber doch höhere Leiterbahnendichte hin. Wenn man den Fräser zu tief einstellt, kann das Leiterbahnen beschädigen. Stellt man ihn nicht tief genug, kann das Kurzschlüsse verursachen. Ist das Platinenmaterial nicht ausreichend eben, können beide Probleme auftauchen. Außerdem ist zu beachten, dass das Fräsen viel Kupfer übriglässt. Das Löten ist dann etwas schwieriger (auch hier fehlt die Lötstoppmaske). Nicht unwichtig ist der gefräste Abstand bei hohen Spannungen wie bei Netzanschlüssen. Auch Leiterbahnen mit hoher Impedanz können Probleme machen. Es ist zwar möglich, unerwünschtes Kupfer wegzufräsen, doch

Kleines Glossar der Platinenentwicklung



Auto-Router: Heiliger Gral der Software-Entwickler für Platinen-Software
Bottom: Der unterste Layer einer Platine, oft als Lötseite bezeichnet
Class: Auflösung oder Dichte einer Platine; je höher der Wert, desto kleiner die erlaubten Elemente und Abstände
Clearance: Abstand zwischen zwei oder mehr Platinenelementen
DRC: Design Rule Check; überprüft die Einhaltung von Design-Parametern wie minimale Bahnbreite, Bohrdurchmesser oder Abstände zwischen den Pads und so weiter
ERC: Electrical Rule Check; prüft, ob alle Netze verbunden und nicht überlappend sind und ob es keine Kurzschlüsse durch Kupferflächen gibt
Excellon: Datenformat für CNC-Bohr- und Fräsmaschinen
Fiducial: Passermarke auf einem Film, einer Maske, Platine etc. als Referenz für Kameras, Stencils, Maschinen oder untereinander
Gerber: ASCII-Vektor-Datenformat für zweidimensionale, zweifarbige Bilder
Heat relief: Kupferpfad (Fläche oder Bahn) zur Verhinderung von Wärmeableitung beim Löten
IAR: Inner Annular Ring
Layer: Schicht mit Platinenelementen wie Bahnen, Flächen, Bauteilen, Lacken oder Aufdrucken
Metalized: Siehe Plated
Mil: 1/1000 Zoll
Net (Netz): Eine erwünschte Verbindung zwischen zwei und mehr Pins
OAR: Outer Annular Ring
Pad: Platinenelement zum Anlöten eines Pins
PCB: Platine
PCB element: Objekt auf einem Layer (z.B. eine Platinenumrandung)
Pin: Anschluss eines Bauteils, z.B. ein Pin, Draht, Lötpunkt
Plane: Große Kupferfläche; Power Plane = mit Versorgungsspannung oder Masse verbunden
Plated: Beschichtet mit einem leitfähigen Material (Metall)
Pour (copper): siehe Plane
Push and shove: Routen einer Leiterbahn durch Verschieben von Platinenelementen rund um die Bahn um genug Abstand für die neue Bahn zu haben
Ratsnest, rat's nest: Eine sichtbare Darstellung aller unverbundenen Netze
Resist: Schutz gegen Lot, siehe Solder mask
Routing: Umwandlung von Netzen in Leiterbahnen
Silk screen: Nichtleitender Layer von grafischen Symbolen und Text, häufig weiß, bekannt unter Bestückungsaufdruck
Short: Unerwünschte Verbindung zwischen zwei oder mehr Pins
Solder mask: Maske mit Aussparung für die Punkte, wo Lot hindarf; verhindert nicht nur Kurzschlüsse, sondern auch das Abwandern von Lot vom Pad und damit schlechte Lötverbindungen sowie die schiefe Ausrichtung von Bauteilen
SMD: Surface-Mount Device; Gegenteil eines bedrahteten Bauteils
SMT: Surface-Mount Technology; Bestückungs- bzw. Löttechnik für SMDs
Stack: Stapel von Layern
Stencil: Maske (aus Edelstahl) zum genauen Aufbringen von Lötpaste auf die Pads einer Platine
Terminal: Siehe Pin
THT: Through-Hole Technology; für Bauteile mit Drähten, die man in Löcher steckt
Tombstoning (Grabstein-Effekt): Teilweise oder komplette Anhebung von SMDs mit zwei Pins beim Reflow-Löten
Top: Der oberste Layer einer Platine; auch unter Bestückungsseite bekannt
Trace: Eine (Kupfer-)Verbindung zwischen zwei oder mehr Pins
Track: Siehe Trace
Via: Metallisiertes Loch, Durchkontaktierung zwischen zwei oder mehreren Kupfer-Layern
Via stitching: Die Verbindung von Kupferschichten verschiedener Layer mit Vias zwecks besserem Stromfluss oder einer besseren Wärmeableitung sowie geringerer Impedanz bei HF

dann braucht das Ganze viel mehr Zeit.
Man kann sich eine Platine sogar selber von Hand „dremeln“, wenn man Fräser mit den runden Köpfen wie beim Zahnarzt einsetzt, doch braucht beides viel Erfahrung und Talent.
Natürlich gibt es auch Profis. Ein Pooling-Service fertigt Ihren Prototypen zusammen mit den Platinen anderer Kunden relativ preiswert auf einer Monsterplatine und sägt die Einzelplatinen hinterher auseinander. Man findet hierfür genügend Angebote im Internet. Die Preise und Lieferbedingungen variieren sehr. Ein nennenswerter Preisvergleich kostet daher Zeit. Die Kosten hängen vor allem von der Platinenfläche, der Layer-Anzahl, der Platinen-Art und der Lieferzeit ab. Zusätzliche Optionen verteuern die Sache. Bei manchen Anbietern kann man Optionen abwählen – bei anderen nicht. Hersteller aus weiter Ferne bieten oft die besten Preise, doch dauert die Lieferung dann länger und die Sendung kann auf dem Transport verloren gehen. Manchmal werden Komplettpreise inklusive Porto angegeben – bei anderen Angeboten kommen dagegen Einrichtungs- oder Werkzeugkosten hinzu.
Man muss die Platine nicht zwingend so klein wie möglich machen. Zwar wird sie dann billiger, doch braucht das Layout dann mehr Zeit und eine Reparatur wird erschwert.
Noch ein Hinweis zum Fräsen: Manchmal braucht man nichtkreisförmige Ausbrüche, damit etwas passt oder man Zugang zu etwas hat. Dies verursacht aufgrund anderer Werkzeuge häufig Extrakosten. Wenn der Ausbruch mit dem gleichen Werkzeug wie beim Trennen der Platinen gemacht wird, erheben manche Hersteller keine Zusatzbeträge. Manchmal kann nicht so präzise gefräst werden, wie man das erwartet hätte. Um Enttäuschungen zu vermeiden, sollte man daher vor der Bestellung schauen, was der Hersteller kann und was nicht.

Bauteilplatzierung

Die Bauteile sollten nicht einfach irgendwo und irgendwie platziert werden – auch dann nicht, wenn genug Platz vorhanden ist. Für bessere Übersicht und einfachere Reparatur sollte man sie regelmäßig in Reihen und Spalten anordnen und gleichförmig ausrichten. Auch wenn das für eine einmalige Bestückung unnötig scheint, erleichtert so ein Vorgehen das eigene Leben und das derer, die später mit der Platine zu tun haben.
Aus elektronischen Gründen sollten Bauteile so dicht platziert werden wie möglich. Natürlich muss man dabei auch noch viele andere Kriterien berücksichtigen wie etwa die Strompfade, Übersprechen oder induktive und kapazitive Effekte. Allgemein sollte man Bauteile sorgfältig platzieren. Für eine Audio-Endstufe gelten andere Kriterien als für ein Präzisionsmessgerät.
Auch die verwendete Löttechnik verursacht Beschränkungen, und selbst die Umgebung kann die Bauteilplatzierung beeinflussen. So können mögliche Feuchtigkeit und Staub auf der Platine größere Leiterbahnabstände erzwingen, um Leckströme zu vermeiden. All das hängt natürlich auch noch von den Normen und Standards ab, denen das Gesamtsystem entsprechen muss. Auch wenn das für die meisten Selbstbau-Projekte eher nachrangig ist, gibt es für diese Reglementierungen gute Gründe. Wenn man sich einliest, kann man dabei sogar noch etwas lernen.
Bedrahtete Bauteile bestückt man immer auf der Oberseite, die nicht ohne Grund Bestückungsseite genannt wird. SMDs kommen wenn möglich auch alle auf eine Seite, egal ob unten oder oben. Eine beidseitige Bestückung macht die Sache teurer. Für Einzelstücke oder Kleinserien spielt das vielleicht keine

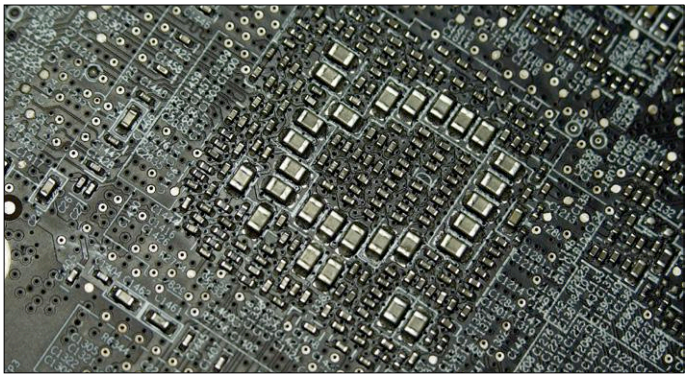


Bild 3. Zur guten Entkopplung von ICs mit vielen Pins können die Entkopplungskondensatoren auch auf die andere Seite der Platine. Die Beschriftungen sollten natürlich an nicht so dicht bestückten Stellen stehen.

große Rolle, doch es ist gute Praxis, sich generell an diese Regel zu halten (**Bild 3**).
Platinen für SMDs benötigen mindestens drei Passermarken als Referenzpunkte. Große SMD-ICs mit vielen Pins und engen Abständen brauchen eventuell zusätzliche Passermarken für eine präzise Ausrichtung bei der maschinellen Bestückung. Bei Selbstbau-Projekten kann man darauf zwar in der Regel verzichten, aber warum sich nicht an diese Praxis gewöhnen? Dieses professionelle Vorgehen kostet ja fast nichts.
Alle gepolten Bauteile wie Dioden und Elkos sollten einheitlich ausgerichtet werden (wenn nicht der Signalverlauf oder andere gute Gründe dagegensprechen). Dieses Vorgehen ist bei Bestückung, Inspektion und Fehlersuche sehr zeitsparend. Die Polarität wird dabei im Bestückungsdruck markiert.

Raster

Man sollte sich soweit wie möglich ans gewählte Raster halten. Für gewöhnlich platziere ich Bauteile und Leiterbahnen zunächst im 50-mil-Raster (1/20“). Bei dicht bestückten Platinen verwende ich 25 mil. Wenn es wirklich sehr eng zugeht, schalte ich auf ein 5-mil-Raster und quetsche so die letzten Leiterbahnen rein. Manchmal reicht auch das nicht und man braucht ein 1-mil-Raster, um die Bauteile gemäß DRC-Kriterien zu platzieren. Statt eines Zoll-Rasters kann man auch ein metrisches Raster verwenden, doch die meisten Bauteile halten sich an mil und Zoll, weshalb es bei metrischen Rastern eher schräge Leiterbahnen gibt.

Pads

Die Drähte und Anschlüsse von Bauteilen werden auf die Pads der Platine gelötet. Meistens ist die Lötstelle auch gleichzeitig die mechanische Befestigung auf der Platine. Die Pads müssen also ausreichend groß sein. Gerade bei hohen oder schweren Bauteilen mit wenigen Pins (große Elkos und Spulen) verhindert dies das versehentliche Abreißen. Die Kupferschicht ist ja nur auf die Platine geklebt und diese Klebestelle kann durch die Hitze beim Löten weich werden und sich lösen.
Für SMDs nimmt man normalerweise längliche Pads. Bleifreies Lot verläuft nicht so gut wie das klassische Sn/Pb 60/40. Das gilt besonders für die Ecken, weshalb man besser Pads mit abgerundeten Ecken wählt. SMD-Pads müssen groß genug sein, um genug Lötpaste für die Pins aufnehmen zu können. Im Datenblatt zum Bauteil finden sich Angaben hierzu.

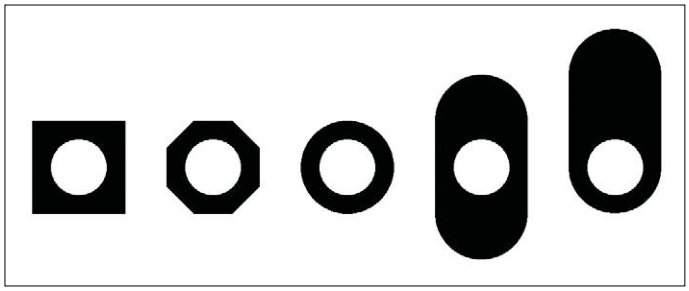


Bild 4. Eine Auswahl an THT-Pads für bedrahtete Bauteile in Eagle.

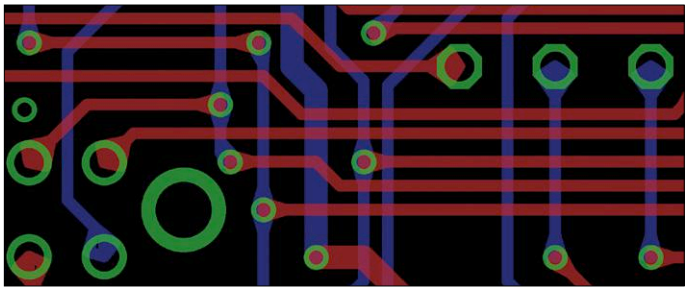


Bild 5. Sich verjüngende „Tapered Pads“ in Eagle, erstellt mit dem „Teardrop Script“ (ULP).

Pads für bedrahtete Bauteile können unterschiedliche Formen haben (**Bild 4**). Rechteckige Pads zeigen häufig Pin 1 eines Bauteils an. Gebräuchlich sind auch achteckige Pads. Eigentlich sind aber runde Pads optimal, denn so wird die größte Kupferfläche bei kleinster Platzverschwendung genutzt, was der Haltbarkeit sowie der Wärmeableitung zugutekommt und große Bahnabstände möglich macht.
Es gibt auch tränenförmige oder spitz zulaufende Pads, die nahtlos in Leiterbahnen münden (**Bild 5**). Verstärkte Übergänge von Pad zu Leiterbahn ergeben nicht nur einen schönen Retro-Touch, sondern helfen auch bei biegbaren, flexiblen Platinen der Belastung zu widerstehen. Nicht jede Layout-Software kann damit aufwarten.
Dann gibt es noch den Punkt Wärmeableitung (**Bild 6**). Wenn ein Pin auf einer sehr breiten Leiterbahn oder Kupferfläche verlötet werden muss, wird durch das Kupfer viel Wärme abgeleitet, was das Löten erschwert. Um dies zu vermeiden, werden die Pads dann nur mit (typisch vier) schmäleren Kupferpfaden elektrisch angebunden bzw. mit Freiflächen thermisch isoliert. Beim Reflow-Löten ist das weniger problematisch, da ja die ganze Platine erhitzt wird. Aber dort kann besonders bei Bauteilen mit zwei Pins wie Widerständen oder Kondensatoren ein sogenannter Grabsteineffekt auftauchen: Bei unterschiedlichen thermischen Massen der Pads können sich die SMDs beim Reflow-Prozess schief stellen. Hier würden dann thermische Isolationstechniken helfen.

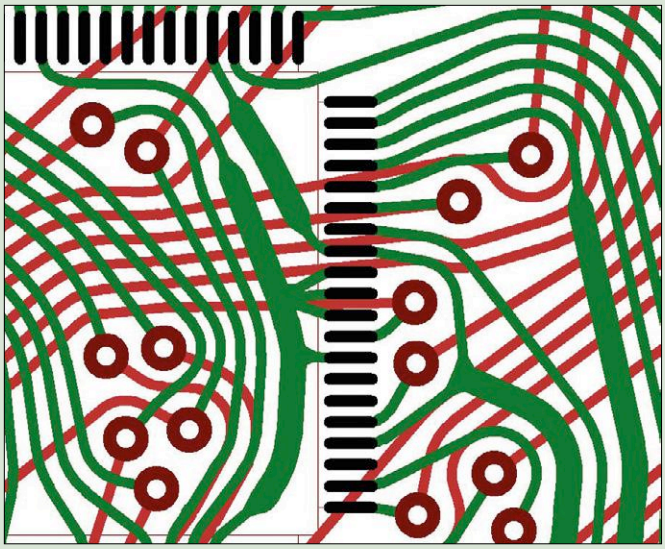
Löcher

Pads haben oft Löcher - die für Vias auf jeden Fall. Befestigungslöcher sind innen nicht beschichtet – Löcher in Pads und Vias hingegen schon. Man gibt den Durchmesser des beschichteten Lochs an. Der Hersteller ist dann dafür zuständig, dass der Lochdurchmesser zum Schluss stimmt. Bei einem beschichteten Loch muss daher um die Metallschichten größer gebohrt werden. Bohren ist aber nur begrenzt genau und die Bohrung kann auch mangelhaft zentriert sein (**Bild 7**). Man

Automatisches oder manuelles Routen?

Die Hersteller von Platinen-CAD-Tools und viele Ingenieure haben schon sehr viel Zeit mit der Entwicklung und Verbesserung automatischer Router verbracht. Bislang war ich noch nicht ein einziges Mal mit dem Ergebnis zufrieden. Ich habe auch noch niemand getroffen, dem es anders ginge. Vielleicht habe ich auch noch nie Spitzen-Software auf dem PC gehabt; aber alles, was ich bisher ausprobiert habe, ist gecrasht oder hat die Arbeit fünf Minuten nach dem Verlassen des Büros eingestellt. Kein Tool konnte 100 % einer Platine routen. Vielleicht waren sie aber auch nur zu kompliziert einzustellen? Auf jeden Fall ziehe ich es vor, meine Platinen von Hand zu routen. Diese Tätigkeit hat für mich etwas Entspannendes.

Ein automatischer Router schafft oft 90 % oder mehr, aber so gut wie nie 100 % einer Platine – außer wenn die Schaltung sehr simpel ist. Und wenn der Autorouter vor 100 % hängen bleibt, hat man leider kein Resultat, das man leicht selbst fertigstellen könnte. Das heißt nicht, dass man die Platine nicht routen könnte, sondern nur, dass der Autorouter jede mögliche weitere Leiterbahn durch seine Vorarbeit blockiert hat. Als Lösung bliebe die Entfernung etlicher Leiterbahnen und deren manuelle Neuverlegung – im Endeffekt ist man schneller, wenn man es gleich selbst macht.
Manche Nutzer verwenden den Autorouter als Kompromiss lediglich dazu, Routing-Probleme aufzuzeigen. Dann werden einige Bauteile anders platziert und die Verbindungen manuell geroutet. Andere verwenden den Autorouter nur auf trivialen Abschnitten einer Platine und räumen hinterher manuell auf.



Dem „topologischen Router“ TopoR von Eremex fehlen bevorzugte Routing-Richtungen zur Optimierung des Platzbedarfs (Quelle: Eremex).

muss also aufpassen, dass die Pads so groß sind, dass nach dem Ätzen und Bohren noch genug Kupfer an den Rändern übrigbleibt. Der OAR (Outer Annular Ring) muss also groß genug sein. Es gibt auch IARs (Inner Annular Rings). Das Wörterbuch meint, dass man unter „annular" etwas Ringförmiges verstehen soll. Also sind das Ring-Ringe. Das erinnert mich an diesen tollen Abba-Song von 1973 (sehr empfehlenswert, siehe [1]). Solche Musik gibt es heute leider nicht mehr. Ebenfalls auffällig: Kleidung, Swenglish, Gitarren, Frisuren... aber ich schweife ab. Ein häufiger Fehler ist eine falsche Angabe eines Bohrdurchmessers. Der Draht des Bauteils passt dann nicht oder verirrt sich im Loch.
Noch ein Punkt zu Vias und Reflow-Löten: Ist ein Via nicht von einer Lötstoppmaske geschützt, dann kann Zinn ins Loch fließen und eventuell ein darüber platziertes Bauteil beschädigen.

Bahnen und Flächen

Sie wissen natürlich, dass man Leiterbahnen so kurz wie möglich hält. Das gilt besonders für HF-Signale, aber auch NF-Signale und stromführende Leiterbahnen profitieren davon. Schließlich verringert sich auch der Platzbedarf. Manche schnellen Signale (oder Gruppen davon) benötigen allerdings eine definierte genaue Länge (und Impedanz), was nicht immer mit der kürzesten Verbindung harmoniert.
Man sollte nicht immer die dünnsten Leiterbahnen verwenden, die im Menü der Software auswählbar sind, denn so dünne Bahnen sind fragil und kosten unter Umständen einen Aufpreis. Das feinste Detail bestimmt die Fertigungsklasse – je höher diese Klasse, desto teurer. Bei Vibrationen können dünne Leiterbahnen Mikrorisse bekommen, was für schlechte oder ausfallende Verbindungen sorgt. Beim Löten von Hand oder bei Reparaturen können sich feine Bahnen bei zu viel Hitze leicht von der Platine lösen. Beim Selberätzen läuft man Gefahr, dass die feinen Bahnen weggefressen werden. Außerdem vertragen dünne Leiter nur wenig Strom. Die Bahnbreite sollte also nach dem zu erwartenden Strom ausgewählt werden. Eine breitere Bahn bleibt zudem kühler. Mit etlichen Online-Rechnern kann man die optimale Bahnbreite bestimmen. Eine überlastete Leiterbahn erwärmt sich oder brennt sogar durch. Auch Vias müssen dem Strom gewachsen sein. Bei Leiterbahnen hilft ein Auftragen von Lötzinn. Das gleiche gilt für Vias: Man kann sie mit Zinn oder einem Kupferdraht füllen. Diese Vias muss man dann in der Lötstoppmaske aussparen. Gängig ist auch, mehrere Vias gewissermaßen parallelzuschalten, was nicht nur die Leitfähigkeit erhöht, sondern auch die Zuverlässigkeit, falls ein Via ausfallen sollte. Schließlich kann man ein Via auch größer machen.
Die aussagekräftige Benennung von Netzen hilft, da so schnell klar wird, mit was für einem Signal man es zu tun hat.
Zu Biegungen: Runde Abbiegungen sind besser, denn scharfe Winkel haben nicht überall die gleiche Leiterbahnbreite, was sich bei hohen Frequenzen in ungenauen Impedanzen oder gar Reflektionen äußern kann. Rechte Winkel sehen nicht gut aus, aber es gibt sie dennoch zuhauf. Selbst bei einer professionellen Platine findet man bei Vias oft, dass das Signal auf der anderen Seite im rechten Winkel abzweigt. Und selbst wenn nicht: Ein Via bedingt ja prinzipiell zwei Winkel zu je 90°. Bahnen mit rechten Winkeln sind prinzipiell länger als welche mit nur 45°. Spitze Ecken sorgen ebenfalls für leichteres Ablösen des Kupfers oder für versteckte Säuretaschen und Unterätzung. Manchmal ist es fast unmöglich, spitze Winkel zu vermeiden, doch meistens geht es problemlos. Ich versuche jedenfalls soweit irgend möglich Winkel mit 45° zu verwenden.
Kupferflächen für die Stromzuführung und insbesondere für Masse sparen nicht nur Verlegearbeit, sondern haben auch bezüglich Störstrahlung Vorteile. Bei mehreren Versorgungsflächen kann es gerade an den Rändern Probleme geben, da z.B. mit Metallschrauben versehene Befestigungslöcher leicht Kurzschlüsse verursachen können. Man sollte zum Schluss immer einen ERC (Electrical Rules Check)

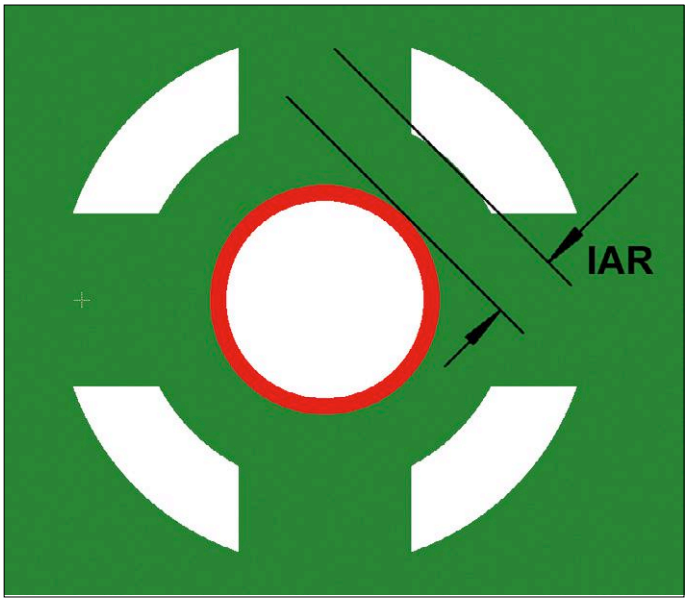


Bild 6. Diese Aussparungen verhindern die Ableitung von zu viel Wärme beim Löten. Hier wird die Stärke des IAR angegeben, obwohl dieser Wert fast der gleiche wie beim OAR ist. Der rote Ring symbolisiert die Beschichtung im Bohrloch.

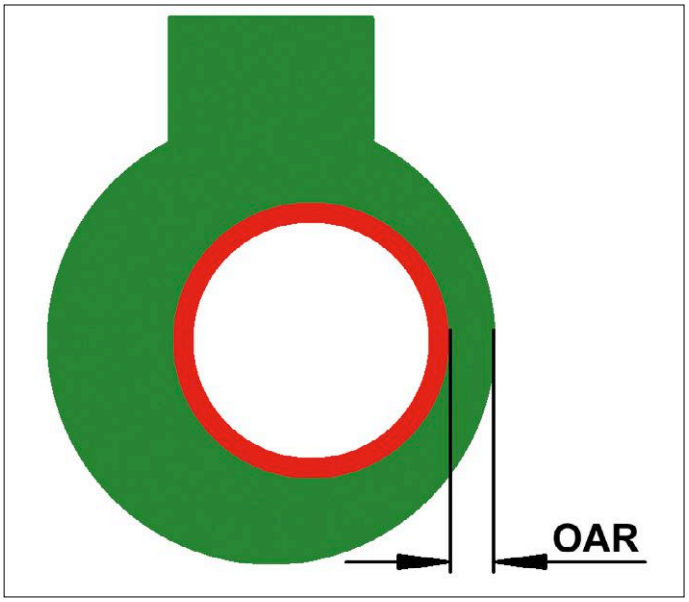


Bild 7. Aufgrund von Fertigungstoleranzen kann eine Bohrung ungenügend zentriert sein. Damit eine Beschichtung (roter Ring) möglich ist, muss das Loch entsprechend größer gebohrt werden. Das beschichtete Loch hat dann den gewünschten Durchmesser. Nach dem Bohren bleibt der AR (Annular Ring) übrig. Der minimale zulässige OAR wird von den Design Rules festgelegt.

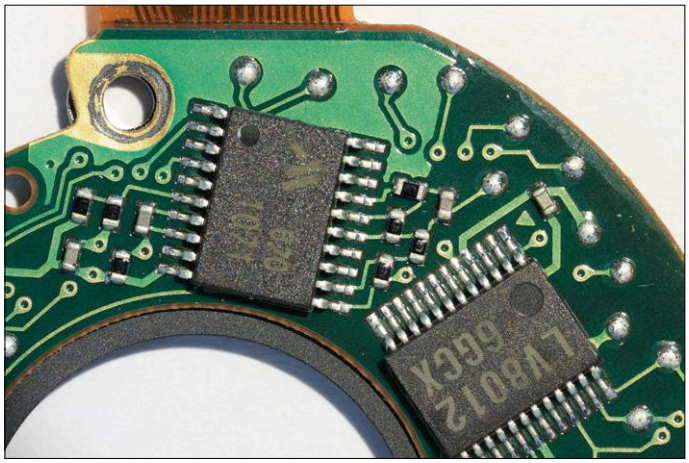


Bild 8. Keine Bauteilbezeichnungen, lediglich die Markierung von Pin 1, Bahnen mit willkürlichen Winkeln und mit Teardrop-Vias. Wenn man weiß, wie man es richtig machen sollte, erscheint die *übliche Praxis* nicht mehr als *gute Praxis*.

durchführen, um sicherzustellen, dass keine Kurzschlüsse oder offene Verbindungen übersehen wurden.
Wenn man Flächen für Netze verwendet, routet man das Netz nicht, sondern überlässt die Verbindung der Kupferfläche. Bei einer segmentierten Fläche sollten deren Verbindungen nicht so dünn sein, dass man beim Selbstätzen Gefahr läuft, diese zu unterbrechen. Um dies zu verhindern, startet man mit einem hohen Wert für die „copper pour clearance“ und schaut, wie die Kupferfüllung aussieht. Für eine verbesserte Füllung kann man dann noch Leiterbahnen und Vias verlegen. Wenn man dann zufrieden ist, kann man den Wert der Clearance wieder ein Stück reduzieren.
Flächen haben logischerweise eine Auswirkung auf die Kupferverteilung der Platine. Bei ungleichmäßiger Verteilung kann der Ätzprozess ungleich ausfallen. Außerdem kann sich dann die Platine bei höheren Temperaturen (wie etwa im Automotive-Bereich) verbiegen.

Design Rules

Man sollte immer DRCs und ERCs auf die Platine loslassen, um sicher zu stellen, dass alle Elemente die Regeln einhalten. Gute *Design Rules* helfen bei der Vermeidung von ungewollten Kreuzungen, beim Aufspüren von falschen Bohrdurchmessern, Abständen und so weiter. Also Checks durchführen! Man stelle zudem sicher, dass alle Netze korrekt angeschlossen sind. Man sollte keine Warnung oder Fehlermeldung stehen lassen, wenn sie nicht akzeptierbar ist, denn man selbst hat die Folgen zu tragen. Nach sechs Monaten kann man in der Regel nicht einmal mehr selbst nachvollziehen, ob ein Fehler vernachlässigbar ist oder nicht. Wenn man also eine Warnung drin lässt, sollte man den Grund dafür dokumentieren. Oder man ändert die Rules so, dass die Meldung nicht mehr auftritt, wobei man natürlich auf vernünftige Regeln sowohl für sich selbst als auch für den Hersteller achten sollte.

Bezeichnungen und Bestückungsaufdruck

Man sollte die Symbole aller Bauteile mit Bezeichnungen versehen und sicherstellen, dass sie auch dann noch sichtbar sind, wenn die Bauteile bestückt sind. Ich ziehe es vor, die Bezeichnungen so auszurichten wie die Bauteile, also in Reihen und Spalten (siehe Bild 3). Besonders bei Steckverbindern und wo das sonst nicht leicht ersichtlich ist, sollte Pin 1 markiert sein. Auch die Polarität von Bauteilen wie Kondensatoren und Dioden sollte klar markiert sein (**Bild 8**).
Man sollte keinen Lack des Aufdrucks unter einem Bauteil haben. Das gilt besonders für zweipolige Bauteile, da diese sonst schief und krumm sitzen.
Text sollte lesbar und Label sollten da vorhanden sein, wo für den Anwender nützlich. Die Lücken in der Lötstoppmaske sind etwas größer als die zugehörigen Pads. Der Bestückungsaufdruck wird davon ebenfalls betroffen und gegebenenfalls abgeschnitten, wenn man nicht aufpasst.
Alle Layer sollten lesbaren Text enthalten, damit nicht versehentlich ein Layer gespiegelt wird. Es bietet sich an, die Layer zu benennen oder zu nummerieren, damit die Reihenfolge nicht durcheinandergebracht werden kann.
Ein Board profitiert von einem eindeutigen Namen und einer Revisionsnummer.
Wenn die Rückseite einer Platine bedruckt werden kann, sollte man das nutzen.
Selbstgebraute Platinen haben zwar nur selten eine Lötstoppschicht, aber das sollte kein Grund sein, Text oder Polaritätsmarkierungen zu vergessen. Man kann auch in Kupfer „schreiben“. Dabei sollte man auch auf die Größe achten, damit die Schrift nicht versehentlich weggeätzt wird.

Tests

Gut zugängliche Testpunkte sind bei Funktionsprüfungen eine große Hilfe. Wenn möglich sollte die Elektronik dort spezielle Testsignale bereitstellen. Anschlüsse von Bauteilen sollte man nicht zum Messen zweckentfremden. Ansonsten könnte z.B. die auf einen Pin drückende Prüfspitze eine kalte Lötstelle vorübergehend leitend machen und eine gute Funktion vorgaukeln. Vias sind gute Testpunkte, aber nur dann, wenn man sie im Lötstopplack ausspart.

Ist das alles?

Nach der Lektüre dieses doch langen Artikels denken Sie vielleicht, dass Sie jetzt viel über das Platinenmachen wissen. In Wahrheit wurde aber lediglich die Oberfläche dieser Disziplin gekratzt. Wir haben überhaupt nicht das Thema Leitungen für die Stromversorgung beleuchtet, oder was es bei mehreren Masseflächen zu beachten gibt. Auch Hitzeaspekte beim Löten oder im Betrieb, die Problematik von Störstrahlung, die speziellen Anforderungen schneller Signale oder wie man die Pad-Anordnung für bestimmte Bauteile kreiert wurde hier nicht besprochen.
Das Thema Platinen-Layout ist eine Spezialkombination aus Chemie, Physik, Elektronik, Mechanik und Automation. Interessant ist, dass viele Aspekte der Platinenentwicklung noch nie wirklich detailliert untersucht wurden, sondern sich schlicht auf den gesunden Menschenverstand und bestimmte Annahmen stützen. Gerade im Internet stößt man auf viele Diskussionen zum Thema Platinen. Dort kann man natürlich auch selbst Fragen stellen.

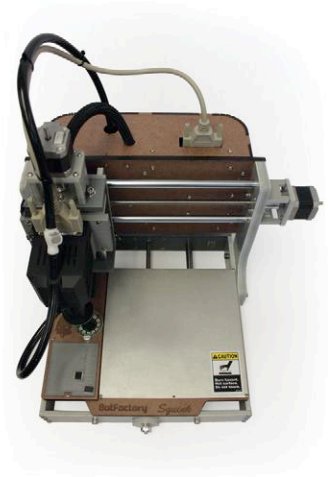
Danke an Malte Fischer für seine nützlichen Hinweise!

(160397)

Weblinks

[1] Abba, https://goo.gl/S2bCpC

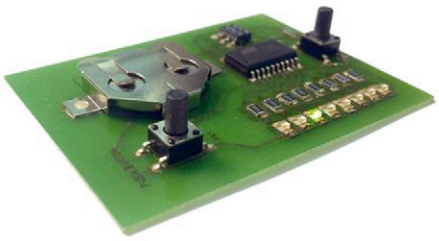
[2] www.elektormagazine.de/160397



Sparen Sie mit uns Platz auf dem Labortisch

Und Zeit und Geld obendrauf! BotFactory Squink ist ein Desktop-Platinenherstellungs-Tool, das in der Lage ist, eine 2-lagige Leiterplatte zu bedrucken, Lötpaste aufzutragen und die Komponenten zu platzieren. Die komplette Herstellung eines Boards ist nun mit einem Gerät möglich.

Squink ist ideal für Entwickler und Dozenten, für kleine und große Firmen. Wenn es um das Prototyping geht, möchte BotFactory, dass Sie sich die unvermeidliche Wartezeit und den hohen Aufwand für eine externe Platinenherstellung und Bestückung sparen.



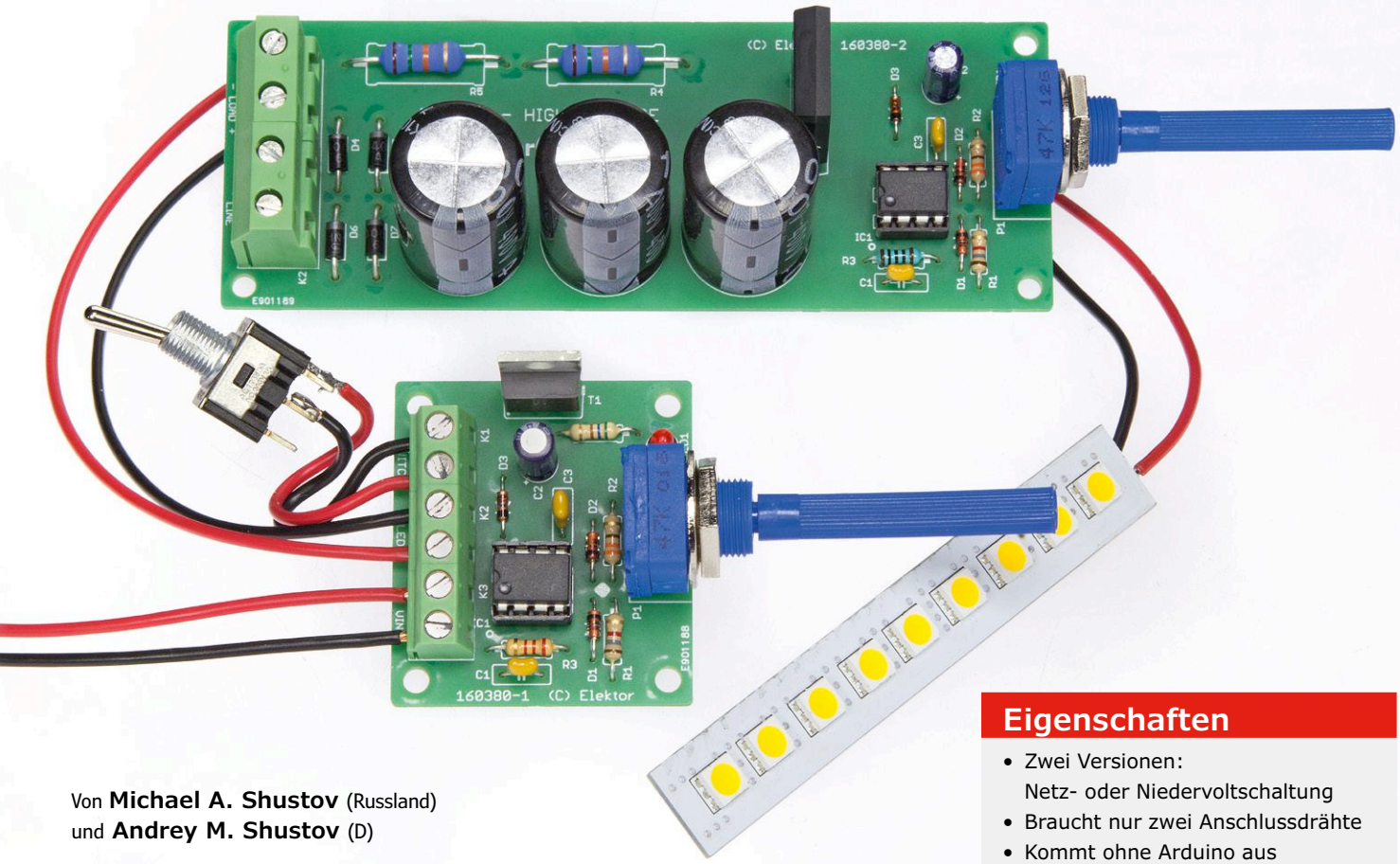
Wir können Sie nicht vor Ihrer Neugier schützen!

Mehr Infos unter www.botfactory.co

BotFactory

2-Draht-Dimmer

Steuert alles: LEDs, Glühlampen und Heizlüfter



Von **Michael A. Shustov** (Russland)
und **Andrey M. Shustov** (D)

Eigenschaften

- Zwei Versionen: Netz- oder Niedervoltschaltung
- Braucht nur zwei Anschlussdrähte
- Kommt ohne Arduino aus

Vor langer Zeit schaltete man einen Rheostaten (heute Poti genannt) einfach in Serie mit der Last, um den fließenden Strom einstellen zu können. Diese einfache Methode hat so ihre Nachteile. Dimmer können das besser.

Zunächst muss der Widerstand eines Potis an die Last und den Einstellbereich angepasst sein. Weiter entsteht im Vorschaltpoti viel sinnlos verheizte Wärme. Schließlich ist so ein „Rheostat" entsprechender Leistung nicht gerade klein und auch nicht preiswert. PWM (Pulse Width Modulation) schneidet besser ab und kann – anders als Phasenanschnitt-Dimmer – nahezu alle Lasten von LEDs über Glühlampen und Heizwendeln bis hin zu Motoren ansteuern.

Normale Dimmer für Beleuchtungszwecke haben in der Regel drei Anschlüsse: zwei für Phase sowie Neutralleiter und der dritte für die Last. Deshalb kann man sie in manchen alten Installationen nicht anstelle des Lichtschalters einsetzen, wenn der Neutralleiter fehlt. Unser Dimmer mit nur zwei Anschlüssen funktioniert in Serie mit der Last und ersetzt daher auch Schalter (siehe **Bild 1**). Der Einstellbereich dieses Zweidraht-Dimmers liegt bei 0,x % bis zu 96 %. In Stellung Minimum fließt nur noch ein kleiner Strom durch die Last. Diese geringe Energieverschwendung ist gegenüber dem Lastverbrauch vernachlässigbar. Der nachfolgende Dimmer kann für häusliche Beleuchtung, Thermostate

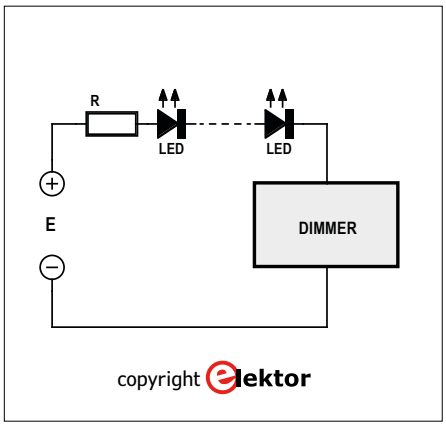


Bild 1. Zweidraht-Dimmer in Serie mit LED(s).

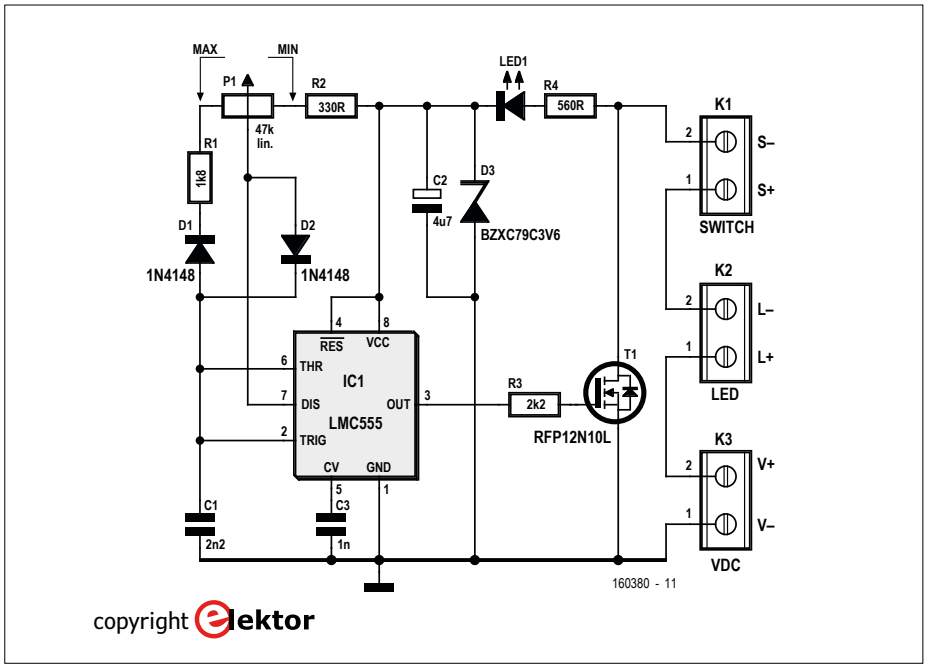


Bild 2. Die Niedervolt-Version des Zweidraht-Dimmers.

oder auch für die Einstellung einer Notbeleuchtung verwendet werden.

Zweidraht-Dimmer für NV-Gleichspannung

Zentrales Bauteil (siehe **Bild 2**) ist der unvermeidliche Timer 555 in Low-Power-CMOS-Ausführung. Er fungiert als Rechteckgenerator mit variablem Tastverhältnis, was nicht unbedingt patentwürdig ist. Der Ausgang von IC1 steuert über den Power-MOSFET T1 den Laststrom (für LEDs etc.) via PWM. Das PWM-Signal hat mit der angegebenen Beschaltung eine flimmerfreie Frequenz von etwa 6 kHz und ist von den R2 (minimaler Impuls) und R1 (minimale Pause) abhängig.

Eigentlich ist die Grundschaltung ein Dreidraht-PWM-Dimmer. Doch wir können auf einen Anschluss verzichten, wenn wir die Schaltung über die Last versorgen. Hierzu braucht es den Speicherkondensator C2, der bei jeder Pause (= T1 sperrt) über R4 und LED1 aufgeladen wird. Die Zener-Diode D3 begrenzt die Spannung. LED1 dient gleichzeitig als Einschaltanzeige.

T1 eignet sich bis 100 V und 12 A, doch gibt es Einschränkungen: $R_{DS(on)}$ = 0,2 Ω. Daher betragen die Verluste bei 2 A schon 0,8 W. Bei größeren Strömen sollte man daher einen adäquaten Kühlkörper vorsehen. Bei Ersatz durch Typen mit kleinerem $R_{DS(on)}$ gilt es zwei Dinge zu beachten: Auch wenn ein CMOS-555 wenig Strom benötigt, steigt er bei größerer Gate-Kapazität von T1. Der Typ RFP12N10L kommt auf nur 0,9 nF. Der zweite Punkt ist die Gate-Schwellspannung: Beim verwendeten Typ beträgt sie 2 V. Bei höheren Werten braucht es eine höhere Spannung von D3. Außerdem ist R4 an die Betriebsspannung anzupassen. 560 Ω eignen sich für 10...18 V. Bei größeren Spannungen sollte durch R4 bei gesperrtem T1 nicht mehr als der zulässige Strom durch LED1 (maximal 25 mA) fließen. Faustregel: Betriebsspannung - (U_{D3} + U_{LED1}) / 25 mA. Bei 24-V-Betrieb und roter LED: 24 V – 5,5 V = 18,5 V. 18,5 V / 25 mA = 740 Ω – passend wäre ein 820-Ω-Widerstand mit einer Belastbarkeit von >0,5 W.

Mit den angegebenen Werten fließen bei 12 V und minimaler PWM etwa 7 mA. Bei maximaler Helligkeit fällt der Strom auf 0,3 mA. Ein Schalter an K1 kann den Dimmer samt Last komplett ausschalten.

Dimmer für Netzspannung

Das Konzept kann man auch auf Netzspannung übertragen. Man hat es entweder mit 230 V (Europa) oder 115 V (USA etc.) Wechselspannung zu tun. Es gibt zwei Hürden:

- T1 muss die höhere Spannung aushalten.
- Wechselspannung muss gleichgerichtet werden.

Eine Einweggleichrichtung reicht nicht aus, denn durch die Halbwellen halbiert sich die Flickerfrequenz angeschlossener Lampen und das kann man unter Umständen deutlich sehen. Außerdem erhöht sich durch Siebung nach Gleichrichtung die effektive Spannung. Aus 230 V Wechsel- werden 325 V Gleichspannung (bzw. 115 V -> 162 V). Das ist für normale Lampen zuviel, wenn der Dimmer voll aufgedreht wird. Man könnte zwar zwei Glühlampen in Reihe schalten, doch das ist nicht praktikabel. Ein Trick für Glühlampen wäre, R1 auf 20 kΩ zu setzen, damit die maximale Ausgangsspannung nur noch 230 V beträgt.

Bild 3 zeigt die Netzversion. Die größte Änderung betrifft die Brückengleichrichtung mit D4...D7 und die Siebung mit C4...C6. Der eingezeichnete Totenkopf symbolisiert, dass Basteln an Netzspannung gefährlich ist.

Es werden drei parallele Sieb-Elkos verwendet, da ein Exemplar mit der Gesamtkapazität teuer und schwer erhältlich ist. Bei 300 µF hat der Strom durch eine 100-W-Glühlampe nur einen Brummanteil von rund 3 % und zudem die doppelte Netzfrequenz, was ein sehr gleichmäßiges Leuchten ergibt. Für stärkere Lasten muss unter Umständen die Kapazität der Siebelkos erhöht werden. Falls keine Lampe, sondern eine Heizung angeschlossen wird, ist die Kapazität nebensächlich.

Für T1 wurde ein Hochspannungs-NPN-Darlington mit hoher Strombelastbarkeit eingesetzt. Aufgrund der Sättigungs-

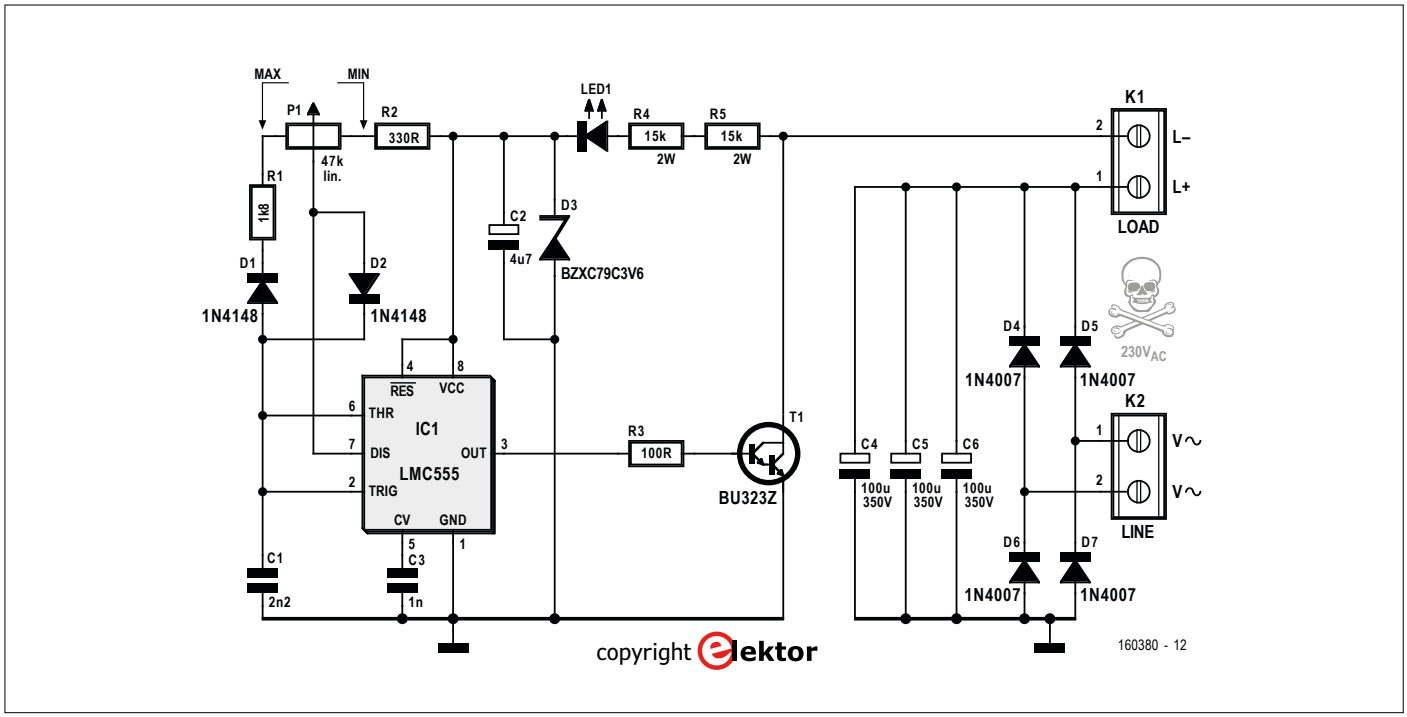


Bild 3. Achtung: Die Netzspannungs-Version des Zweidraht-Dimmers ist gefährlich.

spannung muss er aber bei Lasten über 200 W gekühlt werden. Bei Strömen >1 A müssen die vier Gleichrichter-Dioden durch stärkere Typen ersetzt werden.

Berechnungen

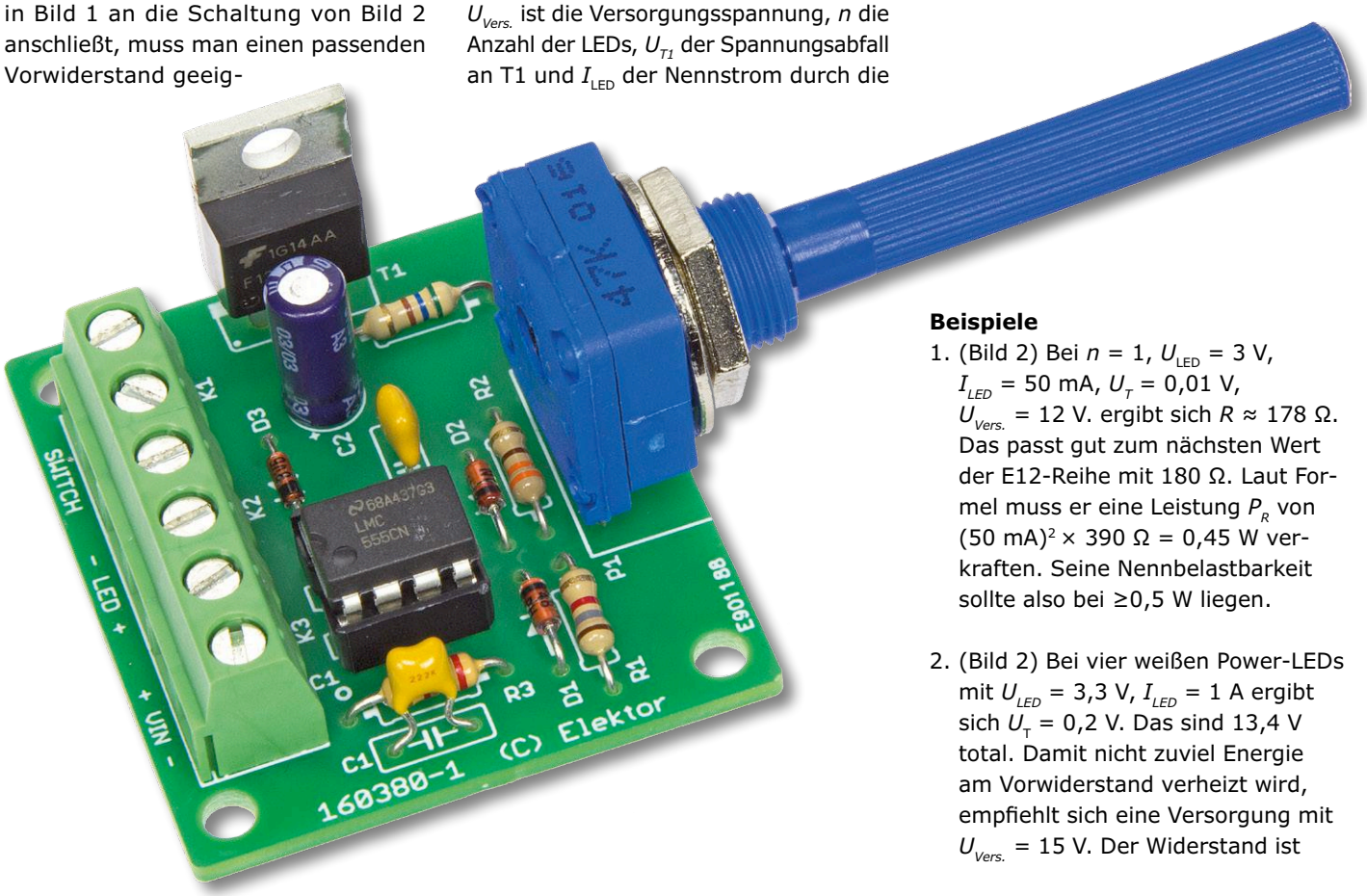
Wenn man eine oder mehrere LEDs wie in Bild 1 an die Schaltung von Bild 2 anschließt, muss man einen passenden Vorwiderstand geeigneter Belastbarkeit vorsehen.

Wie in **Tabelle 1** zu sehen, hängt der Spannungsabfall von der LED-Farbe ab. Der Vorwiderstand berechnet sich zu:

$$R = \frac{U_{SUP} - n \times U_{LED} - U_{T1}}{0.9 \times I_{LED}} \qquad [\Omega]$$

$U_{Vers.}$ ist die Versorgungsspannung, n die Anzahl der LEDs, U_{T1} der Spannungsabfall an T1 und I_{LED} der Nennstrom durch die LED(s). Ein Blick in das LED- und Transistor-Datenblatt ist sinnvoll. Die Verlustleistung des Vorwiderstands errechnet sich zu:

$$P_R = (I_{LED})^2 \times R \qquad [W]$$



Beispiele

1. (Bild 2) Bei n = 1, U_{LED} = 3 V, I_{LED} = 50 mA, U_T = 0,01 V, $U_{Vers.}$ = 12 V. ergibt sich $R \approx 178\ \Omega$. Das passt gut zum nächsten Wert der E12-Reihe mit 180 Ω. Laut Formel muss er eine Leistung P_R von $(50\ mA)^2 \times 390\ \Omega = 0{,}45$ W verkraften. Seine Nennbelastbarkeit sollte also bei ≥0,5 W liegen.

2. (Bild 2) Bei vier weißen Power-LEDs mit U_{LED} = 3,3 V, I_{LED} = 1 A ergibt sich U_T = 0,2 V. Das sind 13,4 V total. Damit nicht zuviel Energie am Vorwiderstand verheizt wird, empfiehlt sich eine Versorgung mit $U_{Vers.}$ = 15 V. Der Widerstand ist

1,6 V / 1 A = 1,6 Ω. Die Abwärme beträgt 1,6 W. Doch 1,6 Ω gibt es nur in der E24-Reihe. Falls das Probleme macht, ersetzt man den 1,6-Ω-Widerstand mit 2 W durch zwei parallele Widerstände mit 3,3 Ω oder zwei serielle mit 0,82 Ω und je 1 W.

3. (Netz, Bild 3, nicht nachmachen!) Große Serienschaltungen z.B. auf LED-Streifen verwenden häufig weiße 100-mW-SMD-LEDs im Format 3528 mit U_{LED} = 3,5 V und I_{LED} = 25 mA. U_{Tmax} des Darlingtons ist 0,6 V. Im 230-V-Netz gilt $U_{Vers.}$ = 325 V. Wenn mindestens 80 % der Leistung an den LEDs anfallen soll, ergibt sich n = 325 V * 0,8 / 3,5 V = 74,3. Also mindestens 75 LEDs in Serie schalten, woran 262,5 V abfallen. Der Vorwiderstand ergibt sich zu (325 V – 262,5 V) / 25 mA = 2,5 kΩ und die Verlustleistung zu 2,5 kΩ * $(25\ mA)^2$ = 1,56 W. Da 2,5 kΩ nicht einmal in der E24-Reihe enthalten ist, empfiehlt sich die Parallelschaltung von 4,7 kΩ und 5,6 kΩ mit je 1 W. Resultat ist ein Vorwiderstand von 2,56 kΩ. An T1 fallen mit 0,6 V * 25 mA nur etwa 15 mW an. Da braucht es keinen Kühlkörper.

Tabelle 1. Flussspannungen von LEDs nach Farbe.

Farbe	Wellenlänge [nm]	*Spannung* [V]
infrarot	>760	<1,6
rot	610 – 760	1,6 – 1,9
orange	590 – 610	1,8 – 2,2
gelb	570 – 590	2,0 – 2,4
grün	500 – 570	2,2 – 2,7
blau	450 – 500	2,6 – 3,3
violett	400 – 450	3,2 – 3,6
ultraviolett	230 – 400	3,5 – 4,2
weiß	–	3,0 – 3,7

Selbst C5 und C6 können bei der Last von rund 8 W entfallen.

Schlusswort

Bitte seien Sie vorsichtig, wenn Sie entgegen der Empfehlung die Schaltung von Bild 3 mit Netzspannung einsetzen. Der Aufbau muss die erforderlichen Sicherheitsabstände aufweisen. Ohne Berechtigung und Erfahrung ist das gefährlich. Mit Spannungen von ≤24 V kann man auch schöne Beleuchtungen realisieren und läuft nicht Gefahr, sich oder Andere zu grillen. Weder die Autoren noch Elektor übernehmen die Verantwortung für das, was Sie mit diesen Informationen und Schaltungen anfangen. Deshalb wird für Bild 3 auch keine Platine angeboten.

Auch bei Bild 2 sollte man auf kurze Leitungen achten (oder gar abschirmen), denn 6 kHz mit harten Flanken machen ordentlich Oberwellen, bei der die Leitungen als Antennen fungieren. Die Amateurfunker in Ihrer Nähe werden es Ihnen danken. ◀

(160380)

Weblink

[1] www.elektormagazine.de/160380

EINKAUFSZETTEL

→ 160380-1
Platine für Niedervolt-Version

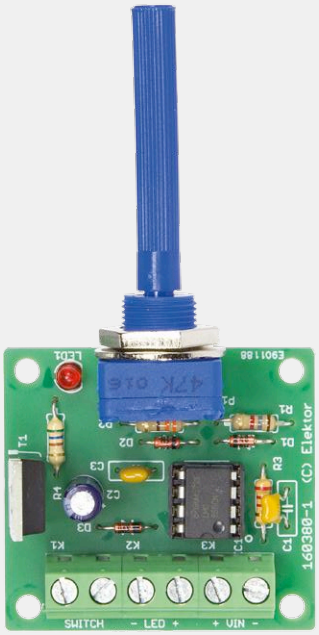
STÜCKLISTE

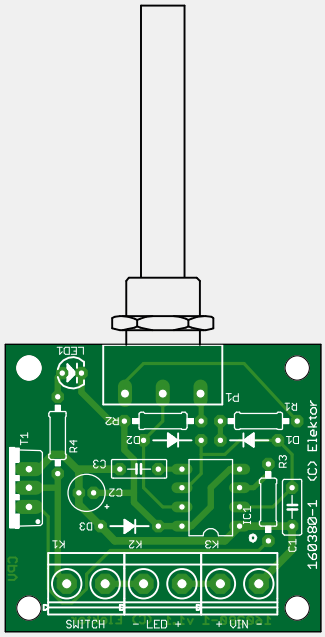
Widerstände:
R1 = 1k8
R2 = 330 k
R3 = 2k2
R4 = 560k
P1 = 47 k, Poti, linear

Kondensatoren:
C1 = 2n2 n, RM 0,2“
C2 = 4µ7 / 10 V, RM 2 mm
C3 = 1 n, RM 0,2“

Halbleiter:
D1, D2 = 1N4148
D3 = BZX79C3V6
IC1 = LMC555CN
LED1 = rot, 3 mm
T1 = RFP12N10L

Außerdem:
K1..K3 = 2-pol. Schraubklemmen für Platinenmontage, RM 0,2”
Platine 160380-1 (siehe Text)





Schaltbarer HF-Abschwächer

Definierte Pegel durch einstellbare Abschwächung

Von **Alfred Rosenkränzer** (D)

Hat man mehr als sehr selten mit Hochfrequenz zu tun, ist es hilfreich, wenn man die Signale auf einfache Weise in diskreten Schritten abschwächen kann. So kann man z.B. das Ausgangssignal eines HF-Generators auf andere Pegel bringen, als es lediglich mit dem Einsteller des Generators möglich wäre. Für einen Abschwächer gibt es viele Anwendungen beim Abgleich oder der Reparatur von Empfängern oder bei Funkgeräten. Leider sind professionelle Abschwächer recht teuer. Hier lohnt sich der Selbstbau, denn kostspielige Elektronik ist nicht erforderlich.

Eigenschaften

- HF-Abschwächer
  mit sechs Schaltern für 0...31 dB
  in 1-dB-Stufen

Manchmal ist es nützlich und nötig, ein HF-Signal definiert abschwächen zu können. Sofern das Signal von einer HF-Quelle stammt, deren Pegel nicht so einfach einzustellen ist, kommt man ohne einen extra Abschwächer kaum weiter. So ein Abschwächer wird üblicherweise einfach in eine HF-Leitung eingeschleift. Nützlich ist das zum Beispiel, wenn man die Empfindlichkeit oder das Kleinsignal-Verhalten eines Empfängers prüfen will: Man schleift den Abschwächer dann einfach in die Antennenleitung ein und erhöht die Abschwächung so weit, bis der Empfang schlecht wird oder die zu untersuchenden Phänomene (wie etwa Rauschen oder AFC-Regelungsartefakte) auftreten. Dem HF-Elektroniker ist sicher auf Anhieb klar, wo er überall einen vernünftigen Abschwächer einsetzen könnte.

Von den bekannten Messgeräteherstellern gibt es solche Abschwächer in guter Qualität (siehe **Bild 1**), allerdings zu einem Preis, den nicht jeder Elektroniker leicht oder gerne auf den Tisch legt. Dabei besteht so ein Abschwächer im Wesentlichen aus nicht mehr als einem Gehäuse, Buchsen, Schaltern und ein paar Widerständen. Mit etwas Sorgfalt ist also der Selbstbau eines guten Abschwächers für gebräuchliche Frequenzbereiche durchaus preiswert möglich.

Abschwächerschaltung

Zur Dämpfung eines Signals unter Beibehaltung des Wellenwiderstandes (Ein- und Ausgangswiderstände sind gleich, aber die Pegel nicht) gibt es prinzipi-

Bild 1. Typische VHF-Abschwächer der Firma HP. Seit langem nicht mehr lieferbar.

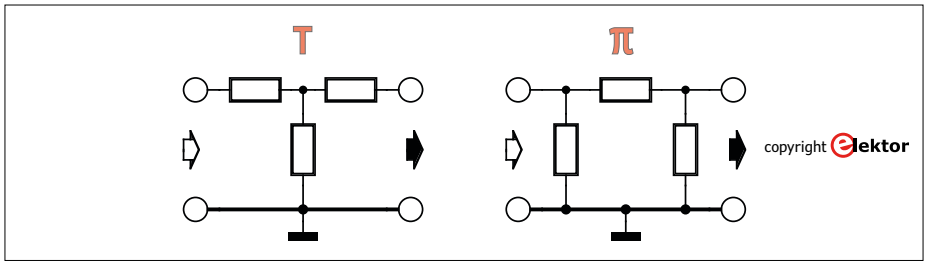


Bild 2. Abschwächer in T- und PI-Grundschaltung.

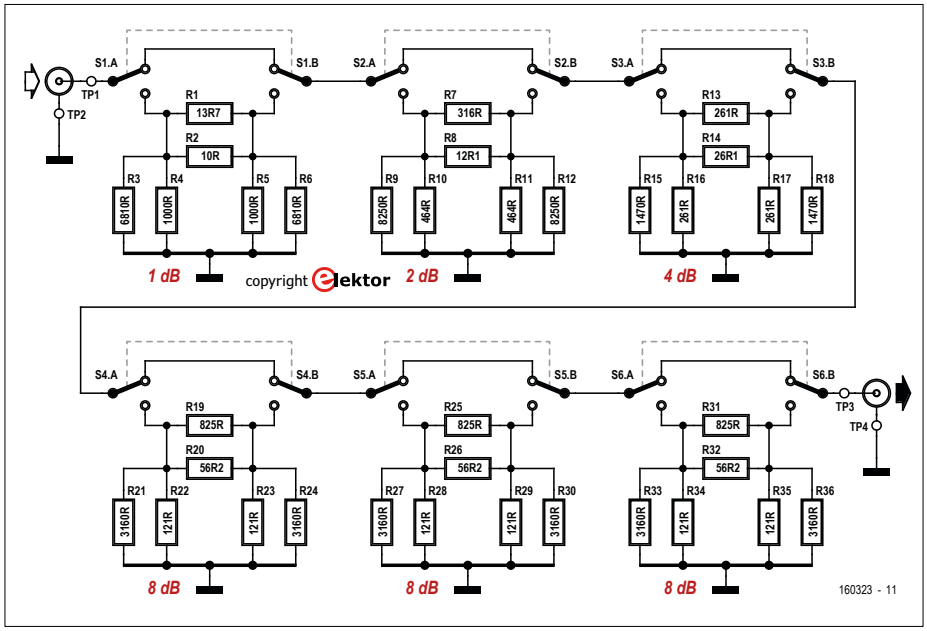


Bild 3. Die Gesamtschaltung des sechsstufigen Abschwächers.

ell zwei gleichwertige Schaltungen aus jeweils drei Widerständen: Die T- und die PI-Grundschaltung (siehe **Bild 2**). Für meine Schaltung habe ich die PI-Variante gewählt, da sie besser im Layout der Platine zu realisieren war. Bei HF ist eine Platine gegenüber einer wilden Verdrahtung vorzuziehen, da bei der Platine die räumlichen Gegebenheiten a) in allen Stufen gleichförmig sind und b) zudem die störenden Induktivitäten und Kapazitäten klein ausfallen.

Bild 3 zeigt die resultierende Schaltung des sechsstufigen Abschwächers. Es sind einfach sechs PI-Abschwächer mit Überbrückung durch doppelpolige Umschalter hintereinandergeschaltet. Die einzige Spezialität liegt darin, dass jeder Widerstand der Grundschaltung durch eine Parallelschaltung von je zwei Widerständen ersetzt ist. Auf diese Weise lässt sich der jeweils errechnete Widerstand trotz der Stufen der E-Reihen möglichst genau realisieren. Insgesamt sind also sechs Schalter und (bis zu) 6 x 3 x 2 = 36 Widerstände vorgesehen.

Die gewünschte Dämpfung wird durch die Dimensionierung der Widerstände erreicht. Theoretisch kann jeder Wert zwischen 0 dB und ∞ dB realisiert werden. Praktisch ist die größte Dämpfung auf etwa 10 dB beschränkt. Bei größeren Dämpfungen stören parasitäre Effekte der Bauteile, der Platine und des Schalters, der sowieso nicht speziell für hohe Frequenzen konstruiert ist. Als kleinster Wert ist 1 dB praktikabel, obwohl natürlich auch Bruchteile eines Dezibels möglich wären. Steht ein Schalter in der oberen Stellung, so wird das Signal ungedämpft weitergeleitet. In der unteren Stellung wird es abgeschwächt. Die Dämpfungswerte der einzelnen Stufen sind dabei voneinander unabhängig, da alle Ein- und Ausgangswiderstände gleich sind. In meinem Aufbau wurden die Dämpfungen 1 | 2 | 4 | 8 | 8 | 8 dB gewählt. Damit lässt sich eine HF-Dämpfung in 1-dB-Schritten (etwa 10 %) im Bereich von 0...31 dB einstellen. Diese Wahl ist für meine typischen Anwendungsfälle praktikabel, doch auch andere Faktoren sind machbar. Sechs Mal 10 dB für 0...60 dB in 10-dB-Schritten oder eine Kombination aus 0,5 | 1 | 2 | 4 | 8 | 8 dB für 0...23,5 dB in 0,5-dB-Schritten sind genauso gut denkbar.

Berechnungen

Zur Berechnung der Widerstände benutze ich das Programm AADE, das man sich unter [1] aus dem Netz ziehen kann. Normalerweise berechnet man mit diesem Programm Komplizierteres wie etwa Filter, doch auch für Abschwächungen taugt es und erspart dabei die Rechnerei. Im Menü *Design – >Attenuator Pad* wählt man zunächst die PI-Struktur und im nächsten Menü den Wellenwiderstand. Prinzipiell könnte man beliebige Werte nehmen, doch heute sind nur noch 50 Ω für Messgeräte und 75 Ω für Antennen und analoge Videosignale gebräuchlich. Die Platine ist für 50 Ω ausgelegt. Durch Änderung der Leiterbahnbreite und Platinendicke kann man aber auch 75 Ω realisieren, wenn man möchte. Entspre-

STÜCKLISTE

Widerstände:

36 x 0603 oder 0805, Werte siehe Text

Außerdem:

S1..S6: doppelpoliger Umschalter für Platinenmontage, TL 46 PO (z.B. Reichelt)

Alu-Druckgussgehäuse, Hammond 1550Z102 (z.B. Reichelt)

2 HF-Buchsen nach Wahl (z.B. BNC für Einlochmontage)

Platine (siehe Text)

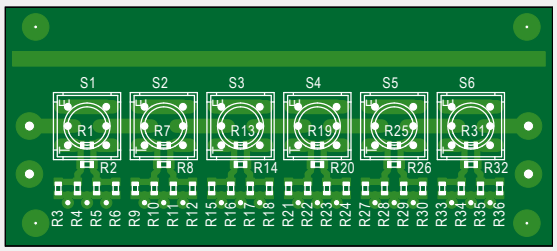


Bild 4. HF-gerechtes Layout der Abschwächer-Platine.

chende Programme sind im Netz zu finden. Wer das vorhat und sich genug auskennt, kann sich das unter [2] verlinkte Tool einmal näher anschauen.
Die gewünschte Dämpfung wird im nächsten Menü eingetragen und man erhält eine Schaltung mit den resultierenden Widerstandswerten. Leider passen die für gewöhnlich nicht genau in das Raster einer E-Reihe. Nicht einmal bei E96 kriegt man immer eine gute Lösung. Damit man auch mit handelsüblichen Werten der E-24-Reihe aus zwei parallelen Widerständen möglichst genau beim von AADE gelieferten Wert landet, kann man Kopfrechnen, seinen Taschenrechner quälen, eine Excel-Tabelle erstellen oder ein kleines Programm schreiben. Noch einfacher aber sind Online-Tools, die einem die ganze Arbeit abnehmen und zu den möglichen Lösungen auch gleich den prozentualen Fehler angeben. Gut geeignet ist z.B. das via [3] nutzbare Tool. Ausreichend sind Fehler <1 %. Wenn Sie sich hier fragen sollten, warum Schaltung und Platine keine Serienschaltungen vorsehen, dann lautet die Antwort, dass die Platine dadurch aufwändiger (und HF-mäßig schlechter) geworden wäre.

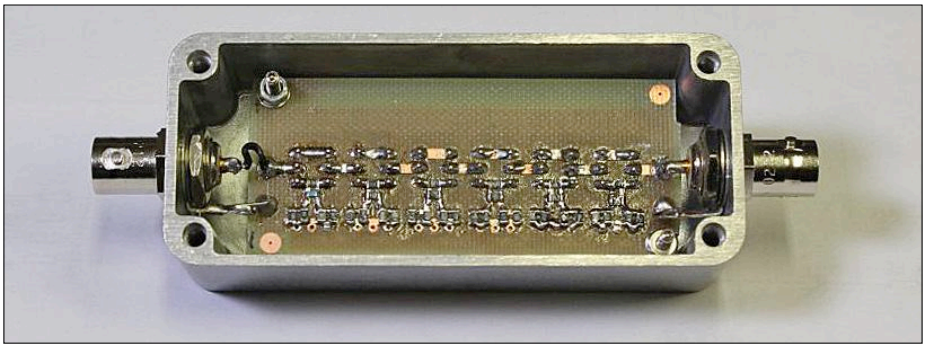


Bild 5. Die Platine eingebaut in das Hammond-Gehäuse.

Bild 6. Der fertig aufgebaute Prototyp.

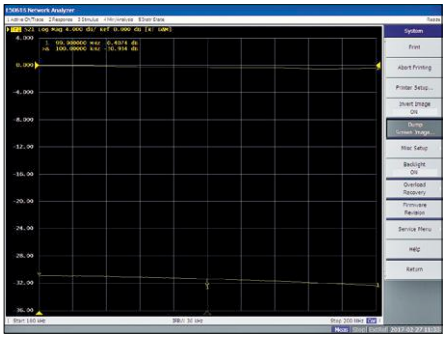


Bild 7. Screenshot der Frequenzgangmessung bei minimaler (0 dB) und maximaler (-31 dB) Abschwächung.

Nichts geht ohne Platine

Die von mir entworfene Platine sieht man in **Bild 4**. Die entsprechenden Dateien stehen im Eagle-Format auf der Elektor-Webseite zu diesem Artikel [4] zum kostenlosen Download zur Verfügung. In der Mitte sind die Schalter angeordnet, direkt darunter jeweils die zugehörigen sechs Widerstände, die mit drei Durchkontaktierungen mit der Masse auf der Unterseite der Platine verbunden sind. Die gewählte Leiterbahnbreite ergibt zusammen mit der 1,0 mm starken FR4-Platine (wichtig!) einen Wellenwiderstand von ca. 50 Ω. Als SMD-Bauform für die Widerstände ist 0603 vorgesehen. Da die Pads recht groß sind, kann man mühelos auch Bauteile in 0805 bestücken. Die Belastbarkeit des Abschwächers hängt direkt von der Belastbarkeit der verwendeten Widerstände ab. Der Abschwächer ist aber eher für kleine Leistungen konzipiert. Man sollte daher besser im Bereich ≤100 mW bleiben. Im Layout wurde übrigens gegenüber dem Prototypen oben eine Teststruktur hinzugefügt, um den Wellenwiderstand messen zu können. Die hierzu vorgesehenen SMA-Buchsen sind optional und werden außer zu dieser Messung nicht benötigt. Die Platine passt genau in ein Aluminium-Druckgussgehäuse der Firma Hammond (siehe Stückliste). Für die Ein- und Ausgänge werden HF-Buchsen (für das Labor in der Regel BNC-Buchsen) verwendet. Beim Bestücken gibt es nichts Spezielles zu beachten, lediglich bei Verwendung von Widerständen im Format 0603 braucht es außer einer guten Pinzette auch eine ruhige Hand. Nach der Bestückung kann man die Platine gut testen, indem man jeweils nur einen Teilabschwächer aktiviert (die restlichen fünf Schalter also auf 0 dB stellt) und dann den Ein- und den Ausgangswiderstand der ganzen Platine misst. Bei der Messung muss man am jeweiligen anderen Pol, also am gegenüberliegenden Ein- oder Ausgang natürlich einen Widerstand von 50 Ω (bzw. 2 x 100 Ω parallel) anschließen. Ergeben sich immer 50 Ω ±<1 %, dann kann man davon ausgehen, dass man keinen Widerstand verwechselt hat. Genau weiß man es aber erst dann, wenn man zusätzlich auch die Abschwächung jeder Stufe misst. Das geht auch mit Gleichspannung, solange man nicht mehr als etwa 5 V über einen 50-Ω-Widerstand einspeist und den Ausgang ebenfalls mit 50 Ω belastet. Aus dem Verhältnis der Ein- und Ausgangspegel ergibt sich dann die reale Dämpfung.

Bild 5 zeigt die in das Hammond-Gehäuse eingebaute Platine und **Bild 6** den fertigen Prototypen. Schön handlich und stabil, nicht wahr? Der Prototyp sieht nicht nur passabel aus, er hat gemessen an der Einfachheit auch ganz ordentliche innere Werte: **Bild 7** zeigt die Frequenzgänge bei einer Abschwächung von 0 dB und dem maximalen Wert von -31 dB. Man sieht, dass der Abschwächer bis etwa 200 MHz ziemlich stabil ist. ⏮

(160323)

Weblinks

[1] http://w1hue.org/filter.html

[2] www.eeweb.com/toolbox/microstrip-impedance

[3] www.qsl.net/in3otd/parallr.html

[4] www.elektormagazine.de/160323

Über den Autor

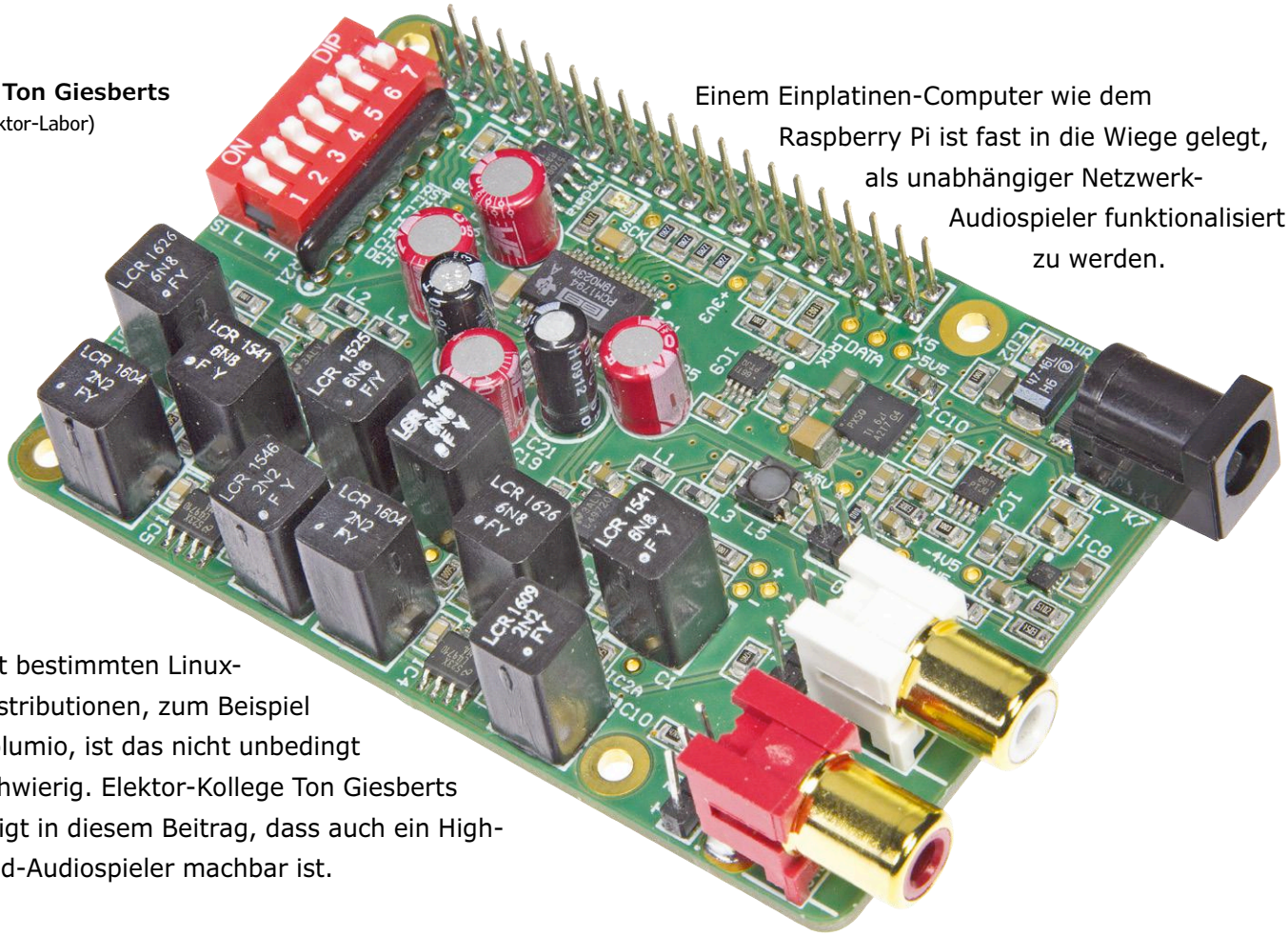
Alfred Rosenkränzer arbeitet seit über 30 Jahren als Entwicklungsingenieur, zu Anfang im Bereich der professionellen Fernsehtechnik. Seit Ende der 1990er Jahre entwickelt er digitale High-Speed-und Analogschaltungen für IC-Tester.

Audio-DAC für RPi

Netzwerk-Audiospieler mit Volumio

Von **Ton Giesberts**
(Elektor-Labor)

Einem Einplatinen-Computer wie dem Raspberry Pi ist fast in die Wiege gelegt, als unabhängiger Netzwerk-Audiospieler funktionalisiert zu werden.



Mit bestimmten Linux-Distributionen, zum Beispiel Volumio, ist das nicht unbedingt schwierig. Elektor-Kollege Ton Giesberts zeigt in diesem Beitrag, dass auch ein High-end-Audiospieler machbar ist.

Das Marktangebot an tragbaren, unabhängigen Netzwerk-Audiospielern, am liebsten ausgestattet mit einem Touchscreen, ist nicht gerade überwältigend. Meistens läuft die Suche auf ein ausgewachsenes Gerät einer „Nobelmarke“ hinaus, zu dem ein ebenso nobler Preis gehört. Doch trotz des Preises fehlt fast immer der Touchscreen. Wenn etwas Kompaktes und Tragbares auf der Wunschliste steht, ist ein solches Gerät sicher nicht das Gesuchte. Das fehlende Angebot legt den Gedanken nahe, den Netzwerk-Audiospieler selbst zu bauen. Eine solide Basis ist der Raspberry Pi: Für ihn sind inzwischen DACs erhältlich, die einen hohen technischen Standard bieten. Doch für Elektor ist ein „hoher Standard“ nicht hoch genug: Wenn wir unseren Netzwerk-Audiospieler schon selbst bauen müssen, können wir einen High-end-DAC für den RPi mit hochqualitativen Bauelementen konstruieren.

Projekt im Überblick

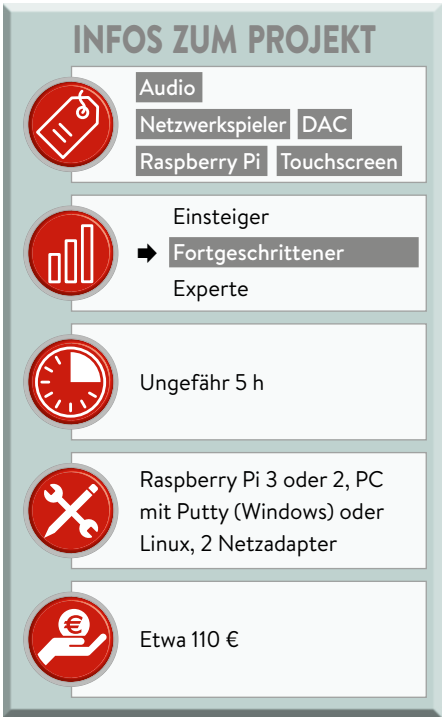
Integrierte Digital-Analog-Wandler (DACs) von Texas Instruments, vormals Burr Brown, halten wir für die hochwertigsten Produkte, die der Markt in diesem Segment bietet. Einige Elektor-Projekte konnten das bereits in der Vergangenheit unter Beweis stellen. Auch bei unserem Netzwerk-Audiospieler kommt ein Spitzenprodukt von TI zum Einsatz, der DAC PCM1794A. Dieser 24-bit-DAC beherrscht Abtastgeschwindigkeiten bis 200 kHz, ein 8-fach-Oversampling-Filter ist integriert. Der Baustein im 28-Pin-SSOP-Gehäuse kann in jedem Detail mit hervorragenden Eigenschaften glänzen. Genannt seien nur der dynamische Bereich 132 dB (9 VRMS, Mono) bis 127 dB (2 VRMS, Stereo), das Verzerrungsmaß 0,0004 % sowie die symmetrischen Stromausgänge, die 7,8 mA_{ss} liefern. Der PCM1794A ist für die gängigen Datenformate ausgelegt: Standard, Left-justified und I2S (Inter IC Sound, wichtig für dieses Projekt). Obwohl der digitale Teil des PCM1794A mit 3,3 V arbeitet, sind die digitalen Eingänge gegen 5-V-Signale spannungsfest. Die vollständigen technischen Daten gehen aus dem Datenblatt von TI [1] hervor.

Um aus dem symmetrischen Ausgangsstrom des DAC eine unsymmetrische Spannung zu bilden, ist für jeden Kanal ein Strom-Spannung-Wandler (I/V) nötig, gefolgt von einem verstärkenden Filter, der die symmetrische Spannung in eine unsymmetrische Spannung umsetzt.
Die vorstehenden Überlegungen führen zum Funktionsschema in Bild 1, mit folgender Liste der Projekt-Zutaten: Basis ist ein Raspberry Pi Modell 3 (oder 2), er wird von dem Audio-DAC sowie einem Waveshare 3,5"-Touchscreen ergänzt. Die Software besteht aus dem Betriebssystem Raspbian für den RPi, Volumio oder eventuell Mood Player sowie den Treibern für den Waveshare Touchscreen.

Betriebsspannungen

Der Digital-Analog-Wandler PCM1794A (IC1, detaillierte Schaltung in Bild 2) benötigt zwei Betriebsspannungen: +3,3 V für den digitalen Teil und +5 V für den analogen Teil. Im Prinzip könnten dazu die Spannungen herangezogen werden, die am GPIO-Steckverbinder des RPi anliegen. Diese Spannungen sind jedoch so stark von Störsignalen überlagert, dass von den High-end-Eigenschaften des Audioplayers kaum etwas übrig bleiben würde. Deshalb müssen die Betriebsspannungen +3,3 V und +5 V unabhängigen Quellen entnommen werden. Alternativ wäre es theoretisch möglich, den RPi an die DAC-Betriebsspannungen zu hängen, doch auch das würde Qualitätseinbußen nach sich ziehen. Ein Nachteil der getrennten Stromversorgung ist der Umstand, dass zwei Netzteile vorhanden sein müssen. Damit nicht noch ein drittes Netzteil hinzukommt, das die negative Betriebsspannung für die I/V-Wandler und das analoge Ausgangsfilter liefert, leitet ein Inverter die negative Spannung aus der Spannung +5 V ab.
Bei genauem Hinsehen fällt auf, dass die analoge Betriebsspannung nicht +5 V, sondern +5,2 V beträgt. Diese etwas höhere Spannung liegt noch im sicheren Bereich des DAC, sie wurde gewählt, damit die unverzerrte Signalspannung des Ausgangsfilters tatsächlich 1 V beträgt. Die ebenfalls gebräuchliche Ausgangsspannung 2 V wäre nur mit Rail-to-Rail-Opamps oder höheren Betriebsspannungen erreichbar. Die meisten Rail-to-Rail-Opamps können jedoch nicht mit den exzellenten Eigenschaften spezieller High-end-Audio-Opamps mithalten.
Zum Bereitstellen der Betriebsspannung +5,2 V kommt der Ultra-low-noise-Low-dropout-Spannungsregler TPS7A4700 zum Einsatz. Der maximale Strom beträgt 1 A bei einem Spannungsabfall von nur 307 mV. Die Ausgangsspannung wird eingestellt, indem bestimmte Pins, die an internen Widerständen liegen, mit Masse verbunden werden. Den Pins sind unterschiedliche Spannungswerte zugeordnet, sie werden zur Ausgangsspannung addiert. Die minimale Ausgangsspannung ist identisch mit der internen Referenzspannung 1,4 V. Für die Ausgangsspannung +5,2 V sind die Pins 6, 10 und 11 an Masse zu legen, bei dieser Konfiguration gilt

$$V_{out} = 1,4V + 3,2V + 0,4V + 0,2V = 5,2V$$



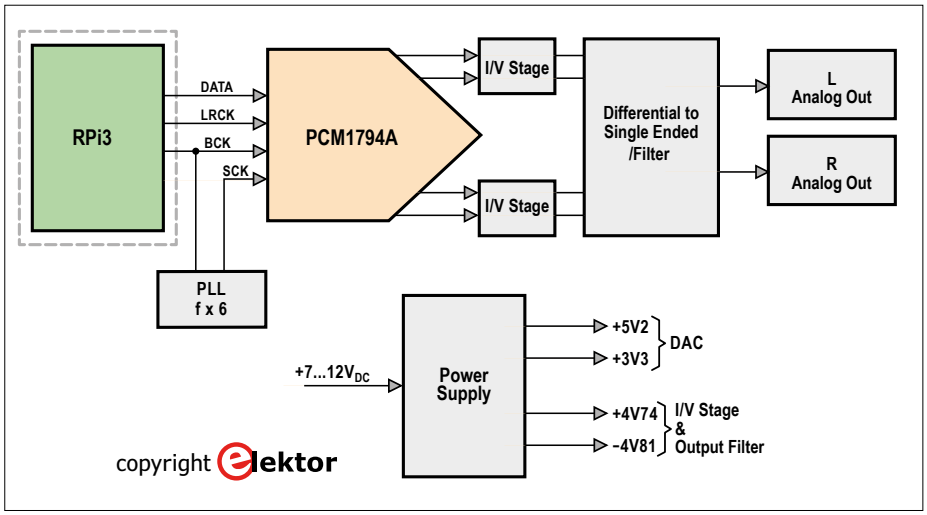


Bild 1. Funktionsschema des Audio-DACs für den Raspberry Pi.

Andere Spannungen

Die positive Betriebsspannung der analogen Ausgangsstufe ist auf +4,74 V eingestellt, sie wird mit einem linearen Ultra-low-noise-Spannungsregler TPS7A4901 (IC7) aus der Spannung +5,2 V abgeleitet. Der exakte Wert der Ausgangsspannung hängt vom Spannungsteiler R28/R29 ab:

$$V_{out} = \left(\frac{R28}{R29} + 1\right) \times 1,185V$$

Der Regler für die Spannung +3,3 V (IC9) ist vom gleichen Typ, hier ist Spannungsteiler R32/R33 für das Einstellen der Ausgangsspannung zuständig.
Die negative Betriebsspannung liefert ein Low-noise-Inverter LM27761 (IC8). Er invertiert die Eingangsspannung mithilfe geschalteter Kondensatoren, aus der invertierten Spannung formt der integrierte lineare Low-noise-Regler die stabilisierte negative Ausgangsspannung. Diese Spannung wird mit Spannungsteiler R30/R31 auf –4,81 V eingestellt:

$$V_{out} = -\left(\frac{R30}{R31} + 1\right) \times 1,22V$$

Für die Ausgangsstufen wurden diese Spannungswerte gewählt (statt des naheliegenden Werts ±4,5 V), um die unverzerrte maximale Ausgangsspannung 1 V (0 dB) zu gewährleisten.
Der Anschluss der Betriebsspannung (K7)

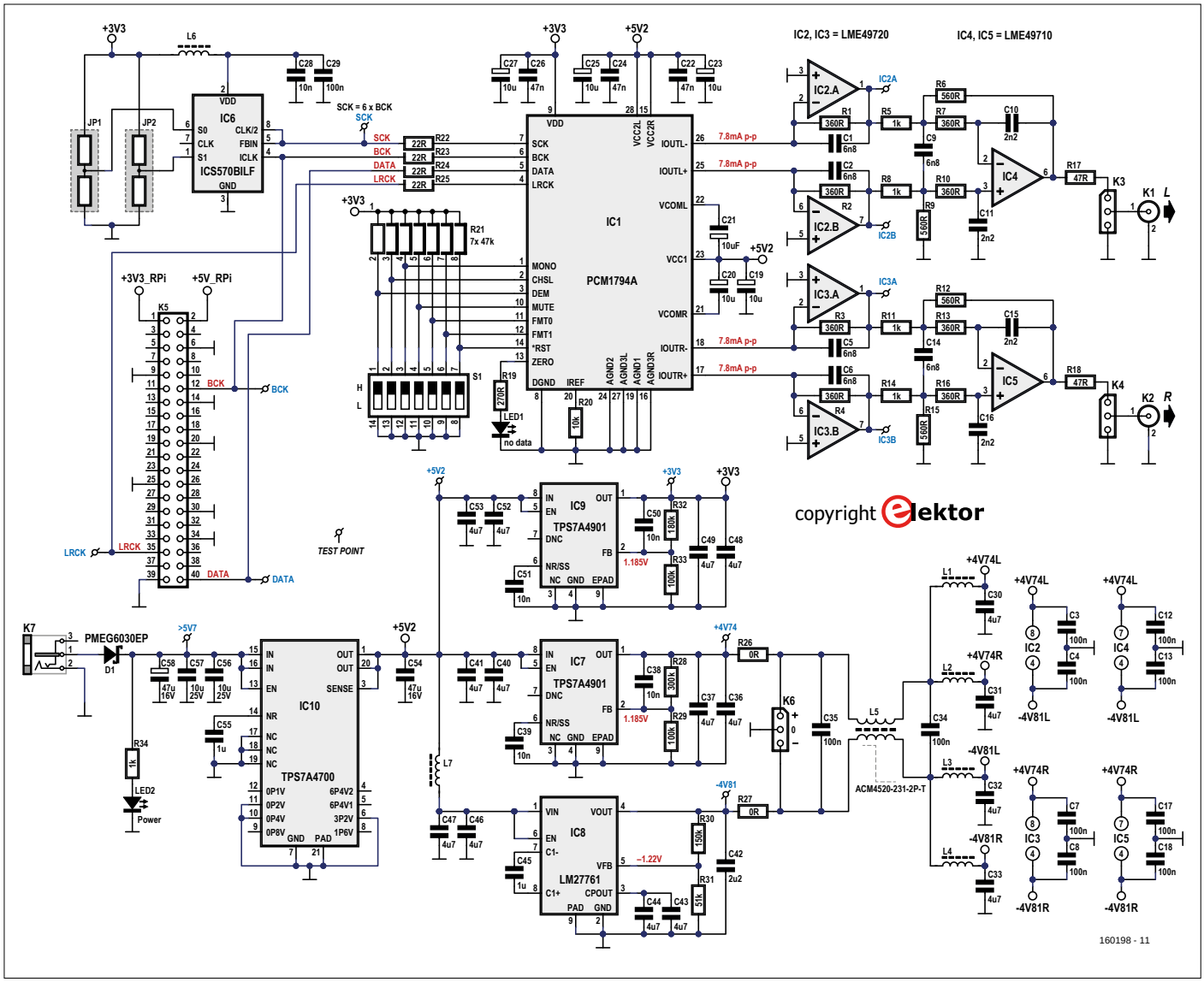


Bild 2. Der DAC-Baustein (IC1) ist ein Produkt von Texas Instruments.

wird von der Schottky-Diode D1 gegen Verpolung geschützt. Der Spannungsabfall an der Schottky-Diode beträgt nur 0,3 V. Die Stromaufnahme des Audio-DAC hängt von der Samplefrequenz ab, Auskunft darüber gibt Tabelle 2.

Master Clock

Damit wir einen Audio-DAC verwirklichen können, wird das Betriebssystem Raspian für einen HiFiBerry DAC konfiguriert. Wenn das geschehen ist, liegen die I2S-Signale an der GPIO-Kontaktleiste an. Die Signale heißen Continuous Serial Clock (SCK) oder Bitclock (BCLK/BCK), Word Select (WS) oder Left/Right Clock (LRCLK/LRCK) und Serial Data (SD oder SDATA).

Für die Synchronisation ist außerdem das Signal Master Clock nötig, es gehört jedoch nicht zu den I2S-Signalen. Ein solches Signal fehlt auf der GPIO-Kontaktleiste, so dass es von der DAC-Schaltung generiert werden muss. Zu diesem Zweck ist ein Multiplizierer mit integrierter PLL hinzugefügt, ein ICS570BILF (IC6). Dieser Baustein arbeitet mit einem Zero Delay Buffer, so dass die Vorderflanken des Eingangssignals und Ausgangssignals perfekt gleichlaufen.

Der Multiplikator lässt sich über zwei Tristate-Eingänge (S0/S1) von 0,5 bis 32 einstellen. Der Frequenzbereich des Ausgangssignals kann laut Datenblatt zwischen 10 MHz und 170 MHz liegen. Hier ist der Multiplikator 6 eingestellt (S0 und S1 bleiben offen), und Ausgang CLK/2 liefert das Signal. Dieser Ausgang ist auch mit dem Feedback-Eingang verbunden. Die Konfiguration gewährleistet den optimalen Eingangsbereich 2,5...12,5 MHz beim Ausgangsbereich 15...75 MHz. Auf den ersten Blick mag es so scheinen, dass ein Signal mit der Samplefrequenz 32 kHz nicht abgespielt werden kann, doch das ist nur ein Trugschluss.

Problem

Beim Testen unseres DACs wurden wir mit einem anderen Problem konfrontiert. Zuerst haben wir diverse 32- und 24-bit-Signale mit Samplefrequenzen zwischen 32 kHz und 192 kHz ausprobiert – mit gutem Erfolg. Beim Testen mit 16-bit-Signalen fehlte jedoch das Ausgangssignal! Kurzfassung einer langen Fehlersuche: Offensichtlich unterstützt der PCM1794A zwar 16- und 24-bit Audio, dies gilt jedoch nicht uneingeschränkt für den I2S-Modus. Das Studium des Datenblattes war hier leider nicht ergiebig...

Bedeutet dies, dass der Audiospieler 16-bit-Audio nicht unterstützt? Ja und Nein! Das Abspielen auf direktem Weg

ist nicht möglich. Unsere Version 1.55 von Volumio hält jedoch die Option offen, die Samplefrequenz zu konvertieren, und zwar in drei Qualitäten. Wahrscheinlich wird diese Funktion in ein zukünftiges Update auf Volumio 2 übernommen.

Der D/A-Wandler

Im DAC des Typs PCM1794A (IC1) ist ein 8-faches digitales Oversampling-Filter integriert, der DAC arbeitet mit Samplefrequenzen von 10 kHz bis 200 kHz. Die I2S-Signale gelangen über Widerstände 22 Ω (R22...R25) zum DAC, um hochfrequente Störungen fern zu halten. Bei der Samplefrequenz 192 kHz beträgt die Frequenz der Master Clock immerhin fast 74 MHz. Die Select-Leitungen MONO, CHSL, DEM, MUTE, FMT0, FMT1, RESET sind auf einen 7-fachen DIP-Schalter geführt, so dass Änderungen an den Einstellungen bequem vorgenommen können. Die sieben Widerstände 47 kΩ haben die Funktion von Pullup-Widerständen. Der ZERO-Ausgang (Pin 13) steuert die LED1, sie signalisiert, dass keine Audiodaten vorhanden sind. Zur Entkopplung der Betriebsspannungen wurden Aluminium-Polymer-Kondensatoren verwendet (C19, C23, C25, C27), weil der ESR nur 40 mΩ bei 100 kHz beträgt. Nachteilig ist allerdings der vergleichsweise hohe Leckstrom, der bis

STÜCKLISTE

Widerstände (1 %, 0,125 W, SMD 0805, wenn nicht anders angegeben):

R1,R2,R3,R4,R7,R10,R13,R16 = 360 Ω
R5,R8,R11,R14,R34 = 1 k
R6,R9,R12,R15 = 560 Ω
R17,R18 = 47 Ω, 125 mW
R19 = 270 Ω, 125 mW
R20 = 10 k, 125 mW
R21 = 47 k, Widerstandsnetzwerk 7-fach SIP, 125 mW, 2 %
R22,R23,R24,R25 = 22 Ω
R26,R27 = 0 Ω
R28 = 300 k
R29,R33 = 100 k
R30 = 150 k
R31 = 51 k
R32 = 180 k

Kondensatoren:

C1,C2,C5,C6,C9,C14 = 6n8, 63 V, 1 %, Polystyrol, EXFS/HR 6800PF, ±1 %, LCR Components (wahlweise RM 5 mm oder SMD 0805)
C3,C4,C7,C8,C12,C13,C17,C18,C29,C34,C35 = 100 n, 50 V, 10 %, X7R, SMD 0805
C10,C11,C15,C16 = 2n2, 63 V, 1 %, Polystyrol, EXFS/HR 2200PF ±1%, LCR Components (wahlweise RM 5 mm oder SMD 0805)
C19,C23,C25,C27 = 10 µF, 35 V, 0,04 Ω, d=6,3 mm (max.), RM 2/2,5 mm, 870055673001 (WCAP-PTHR Series), Würth Elektronik
C20,C21 = 10 µF, 63 V, 1,06 Ω, d=6,3 mm (max.), RM 2/2,5mm, UPM1J100MDD, Nichicon
C22,C24,C26 = 47 n, 50 V, 10 %, X7R, SMD 1206
C28 = 10 n, 50 V, 10 %, X7R, SMD 0603
C30,C31,C32,C33,C36,C37,C40,C41,C43,C44,C46,C47,C48,C49,C52,C53 = 4µ7, 25 V, 10 %, X5R, SMD 0805
C38,C39,C50,C51 = 10 n, 50 V, 10 %, X7R, SMD 0805
C42 = 2µ2, 10 V, 10 %, X7R, SMD 0805
C45,C55 = 1 µ, 16 V, 10 %, X7R, SMD 0805
C54 = 47 µF, 16 V, 20 %, X5R, SMD 1210
C56,C57 = 10 µF, 25 V, 10 %, X5R, SMD 1206
C58 = 47 µF, 16 V, 10 %, Tantal, SMD-C (2312), TR3C476K016C0350, Vishay

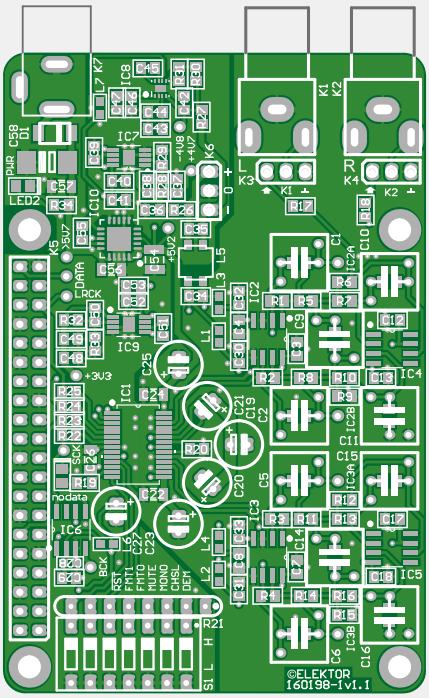


Bild 3. Die Bestückungsseite der Platine.

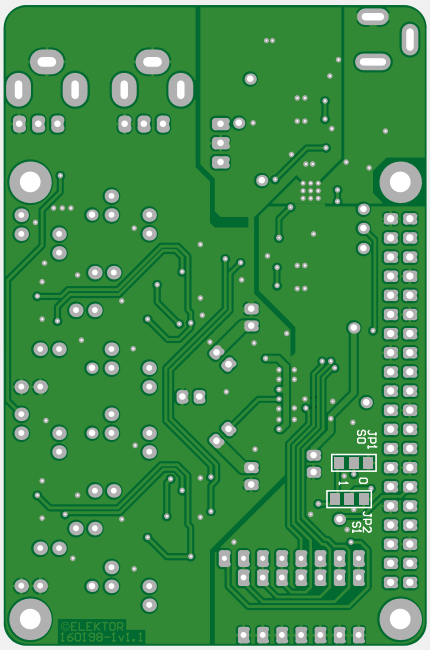


Bild 4. Die Platinenunterseite. Normalerweise können JP1 und JP2 entfallen.

Induktivitäten:

L1,L2,L3,L4,L6,L7 = 600 Ω @ 100 MHz, 0,15 Ω, 1,3 A, SMD 0603, BLM18KG601SN1D, Murata
L5 = ACM4520-231-2P-T, 2 · 0,05 Ω, 230 Ω @ 100 MHz, 2,6 A, SMD, TDK

Halbleiter:

D1 = PMEG6030EP, 60 V, 3 A, SMD SOD-128
LED1,LED2 = LED grün, Low-power, SMD 0805
IC1 = PCM1794ADB, SMD SSOP-28
IC2,IC3 = LME49720MA/NOPB, SMD SOIC-8
IC4,IC5 = LME49710MA, SMD SOIC-8
IC6 = ICS570BILF, SMD, SOIC-8
IC7,IC9 = TPS7A4901DGNT, SMD MSOP-8
IC8 = LM27761DSGT, SMD WSON-8
IC10 = TPS7A4700RGWT, SMD VQFN-20

Außerdem:

K1 = Cinch-Buchse, weiß, Platinenmontage, abgewinkelt, vergoldet, PJRAN1X1U02AUX, Switchcraft
K2 = Cinch-Buchse, rot, Platinenmontage, abgewinkelt, vergoldet, PJRAN1X1U03AUX, Switchcraft
K3,K4,K6 = Stiftkontaktleiste 3-polig, stehend, RM 2,54 mm
K3,K4 = Jumper, RM 2,54 mm
K5 = GPIO-Stapelkontaktleiste 2 · 20-polig, extra hoch
K7 = Stromversorgungsbuchse, 3 A, 1,95 mm, Lumberg NEB 21 R
S1 = DIP-Schalter 7-fach
4 · Abstandsbolzen mit Schraubgewinde M2,5, 17 mm lang
4 · Abstandsbolzen mit Schraubgewinde M2,5, 14 mm lang
8 · Mutter M2,5

Platine 160198-1, V1.1

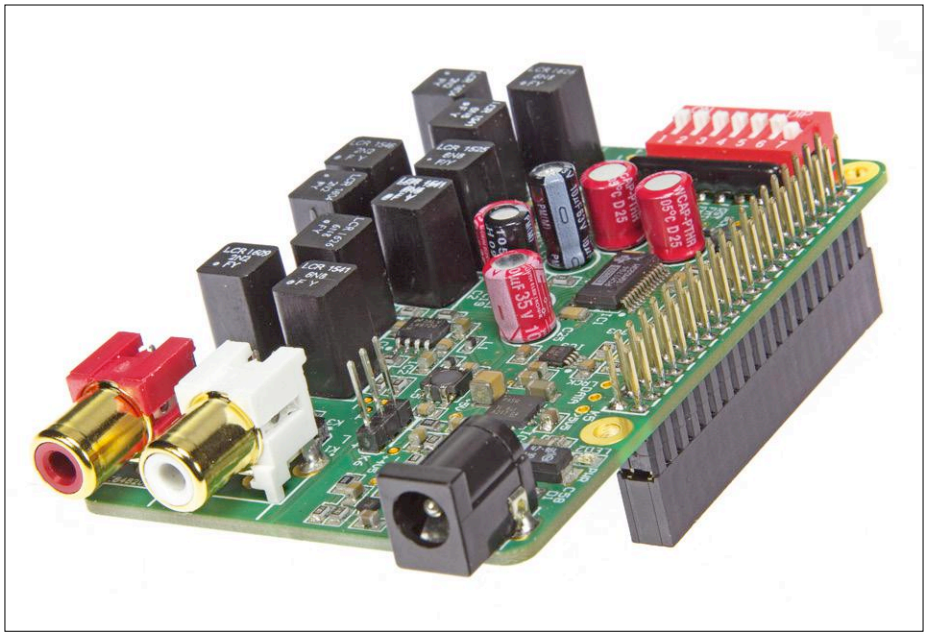


Bild 5. Die GPIO-Kontaktleiste wird auf der Unterseite montiert.

100 µA betragen kann. Da nicht sicher ist, ob dieser Leckstrom Einfluss auf den internen Einstellstrom hat, kamen auch zwei herkömmliche Elektrolytkondensatoren (1,06 Ω bei 100 kHz und 4 µA Leckstrom) zum Einsatz. Auf der Platine ist genügend Platz vorhanden, um mit unterschiedlichen Kondensator-Typen zu experimentieren.

Tabelle 1: Einstellungen von S1		
S1-1	L	Deemphasis für 44,1 kHz ausgeschaltet
S1-2	L	Steil abfallendes digitales Filter
S1-3	L	Mono aus
S1-4	L	Mute aus
S1-5, S1,6	L	I^2S-Modus
S1-7	H	Reset aus

Tabelle 2: Stromaufnahme abhängig von der Samplefrequenz	
Samplefrequenz	Stromaufnahme (K7)
No Data	120 mA
32 kHz	127,5 mA
44,1 kHz	131,7 mA
48 kHz	133 mA
96 kHz	149,5 mA
192 kHz	182,1 mA
(Gemessen mit Labornetzteil 8 V, Ausgänge abgeschlossen mit 10 kΩ)	

Tabelle 3: Verzerrungen				
THD+N				
Samplefrequenz		48 kHz	96 kHz	192 kHz
1 kHz	B = 22 kHz	0,0008 %	0,0009 %	0,0013 %
	B = 80 kHz	0,0028 %	0,0012 %	0,0014 %
7 kHz	B = 22 kHz	0,00095 %	0,0011 %	0,0013 %
	B = 80 kHz	0,003 %	0,0014 %	0,0016 %
IMD				
50 Hz : 7 kHz = 4:1		0,0014 %	0,002 %	0,0036 %

Ausgangsfilter

Auch wenn Abtast-Artefakte größtenteils vom digitalen 8-fach-Oversampling-Filter unterdrückt werden, enthält das DAC-Ausgangssignal noch hochfrequente Signalanteile, die dort nicht hingehören. Diese Signalanteile werden mit einem Rekonstruktionsfilter unterdrückt, einem Butterworth-Filter dritter Ordnung. Die I/V-Wandler IC2/IC3 sind Teil dieses Filters, seine Eckfrequenz liegt bei 64,5 kHz. Bei der Abtastfrequenz 44,1 kHz wird die 8-fache Komponente bei 352,8 kHz um mehr als 40 dB abgeschwächt, und bei der Abtastfrequenz 192 kHz beträgt die Abschwächung mehr als 80 dB. Die gewählte Eckfrequenz ist ein Kompromiss zwischen ausreichender Filterung bei niedrigen Abtastfrequenzen und der NF-Bandbreite des Audiosignals bei hohen Abtastfrequenzen. In diesem Fall beträgt die Filterbandbreite mehr als das Dreifache des hörbaren Frequenzbereichs.

Wegen der symmetrischen Stromausgänge des DACs ist ein externes Filter einsetzbar. Um den symmetrischen Ausgangsstrom in eine unsymmetrische Ausgangsspannung zu konvertieren, ist zuerst ein I/V-Wandler nötig. Der I/V-Wandler ist mit den doppelten Opamps in IC2 und IC3 aufgebaut. Die symmetrischen Ausgangsspannungen von IC2A/IC2B (IC3A/IC3B) gelangen nach IC4 (IC5), dort werden sie in unsymmetrische Ausgangsspannungen umgesetzt. Gleichzeitig haben diese Stufen die Funktion von Filtern zweiter Ordnung. In Kombination mit den I/V-Wandlern entspricht die Filtercharakteristik einem Butterworth-Filter dritter Ordnung. Auf der Platine wird das unsymmetrische Ausgangssignal auf die Steckverbinder K1 und K2 über die Jumper K3 und K4 geführt. Es ist auch möglich, an K3 und K4 ein Stereo-Potentiometer zur Volumeneinstellung anzuschließen.

Der DAC hat keine Volumeneinstellung an Bord. Volumio kann zwar das Volumen über die Software einstellen, doch das geht auf Kosten der Auflösung und damit der Audioqualität. Natürlich wäre dies keine zufriedenstellende Lösung. Deshalb wurde eine Volumeneinstellung auf der Seite der Hardware vorgesehen, damit ist auch die Fernbedienung des Volumens möglich. Die Volumeneinstellung

wird in einer folgenden Elektor-Ausgabe beschrieben.
Ein letztes Detail: Die symmetrische Betriebsspannung für das Filter und die I/V-Wandler werden von der Betriebsspannung der digitalen Stufen mit einer Gleichtaktdrossel und vier einfachen Drosseln entkoppelt, für jeden Kanal separat.

Die Platine

Die DAC-Platine kommt mit einer einzelnen Betriebsspannung 7...8 V aus, denn der Raspberry Pi läuft an der eigenen Spannung 5 V. Die Spannung der DAC-Platine kann zwar im Bereich 7...12 V liegen, doch bei niedrigen Werten setzt der Spannungsregler IC10 weniger Verlustleistung um.
Die Layouts der Platine sind in Bild 3 und Bild 4 wiedergegeben, die Abmessungen sind identisch mit dem RPi. Der siebenfache DIP-Schalter hat seinen Platz an der Seite, die der Stromversorgungsbuchse gegenüber liegt. Beim RPi 3 befindet sich dort die WLAN-Antenne. Die 40-polige GPIO-Kontaktleiste K5 wird auf der Unterseite montiert. Die Fotos zeigen anschaulich, wie die aufgebaute Platine aussieht.
Die auf der Platinenunterseite überstehenden Stifte der Kontaktleisten K1... K4, K6 und K7 müssen möglichst kurz abgekniffen werden, um Kurzschlüsse mit den Metallgehäusen der Netzwerk- und USB-Buchsen auf dem RPi zu vermeiden. Eine zwischen die Platinen geschobene isolierende Scheibe ist ein zusätzlicher Schutz. Bei unserem Labormuster haben wir metallene Abstandsbolzen (2,5 mm, 17 mm lang) mit Schraubgewinde verwendet, um den DAC auf dem RPi zu befestigen. Die Lötinseln für die Kondensatoren des Filters sind so gestaltet, dass unterschiedliche Kondensator-Typen montierbar sind. Gemeint sind die Kondensatoren C1, C2, C5, C6, C9... C11 und C14...C16. Auch Exemplare in der Bauform SMD 0805 können montiert werden, radiale Kondensatoren im Rastermaß 5 mm passen ebenfalls. Mit Blick auf optimale Resultate empfehlen wir den Einsatz von Polystyrol-Typen mit der Toleranz 1 %.
Der Platinenaufdruck gibt bei den ICs die Position von Pin 1 durch einen Markierpunkt an, bei IC7 ist dies jedoch wegen C38 nicht möglich. Auch für die Bezeichnungen der 0805-Widerstände sowie der 0805- und 1206-Kondensatoren ist neben diesen Bauelementen nicht genügend Platz. Die Bezeichnungen stehen innerhalb des rechteckigen Symbols, so dass sie später auf der Platine verdeckt sind. Beim Bestücken der Platine von Hand (Heißluftlötkolben oder Lötofen) ist eine vergrößerte Kopie des Bestückungsaufdrucks hilfreich. Sobald die bedrahteten Komponenten an ihrem Platz sind, können die meisten SMD-0805-Komponenten bei falscher Bestückung nur noch mühsam getauscht werden. Deshalb müssen die bedrahteten Komponenten zuletzt montiert werden.
Die Jumper JP1 und JP2 befinden sich auf der Platinenunterseite, im Normalfall bleiben sie offen (SCK = 6 · BCK). Die Cinch-Buchsen an den Audioausgängen sind Typen des Herstellers Switchcraft.

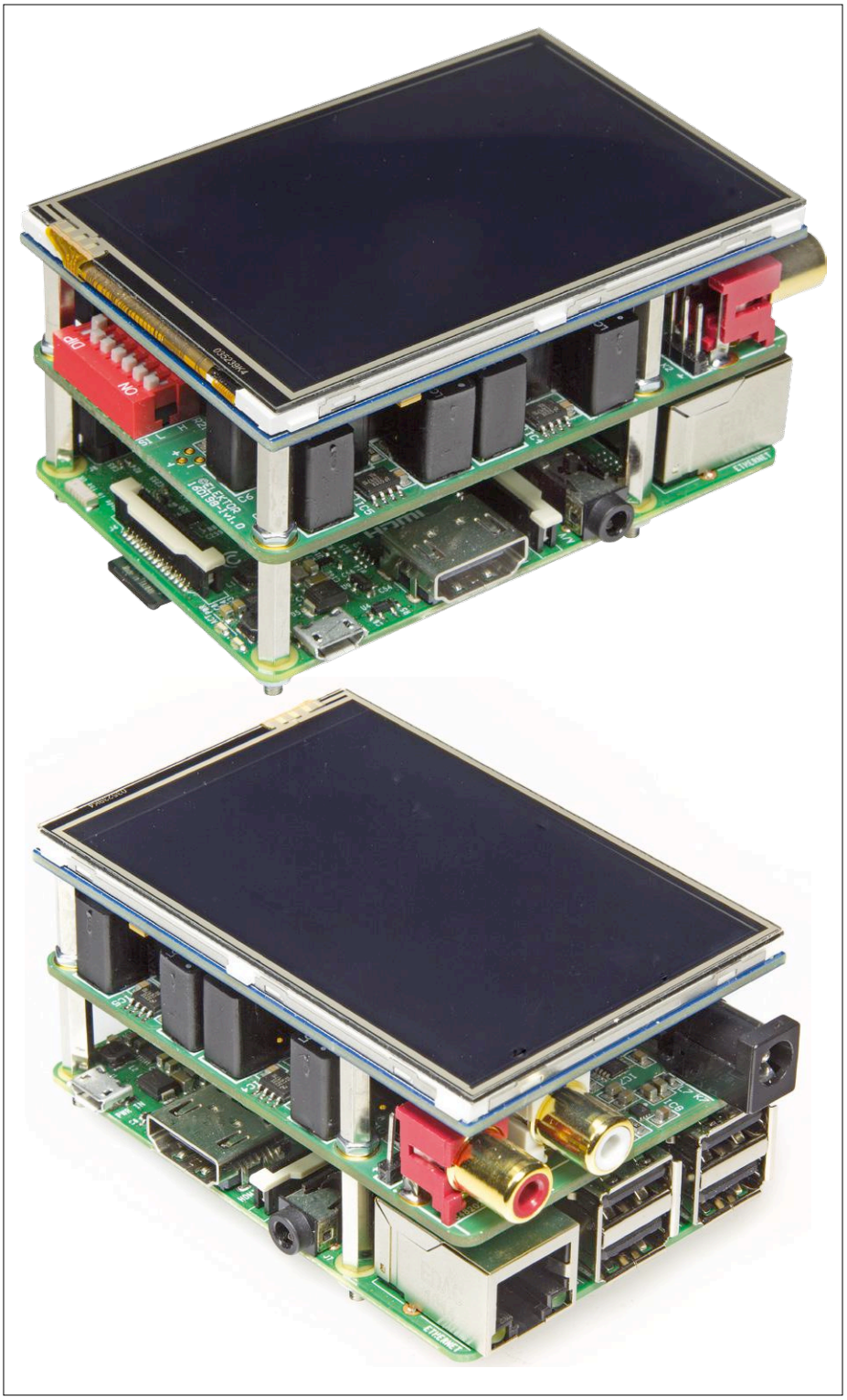


Bild 6. Der betriebsbereite Netzwerk-Audiospieler.

Volumio

Volumio ist eine kostenfreie Open-source-Linux-Distribution, die für die Musikwiedergabe maßgeschneidert wurde. Die Software läuft auf einer Vielzahl großer Systeme, aber auch auf kleinen, preisgünstigen Computern wie dem Raspberry Pi. Nach dem Installieren von Volumio verwandelt sich das System in einen Headless Audiophile Music Player. *Headless* bedeutet, dass der Player über ein anderes System wie Smartphone, Tablet oder Computer bedient werden muss. Das geschieht über die Anwenderschnittstelle von Volumio: Eine Web-App, die auf jedem Browser läuft, erlaubt das unkomplizierte und intuitive Abspielen von Audiodateien. Die Kommunikation zwischen der Web-App und Volumio wird über das lokale Netzwerk abgewickelt.
Auf die Projektseite [2] haben wir eine ausführliche Anleitung zum Installieren von Volumio V1.55 für den RPi gestellt. Damit ist Volumio auch (aber nicht nur!) über den Touchscreen bedienbar. **Bild 7** gibt einen Eindruck von der Bedienung über den Touchscreen (links) und den PC (rechts).

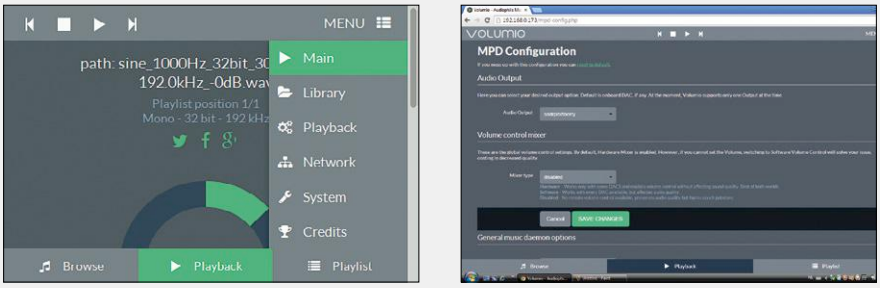
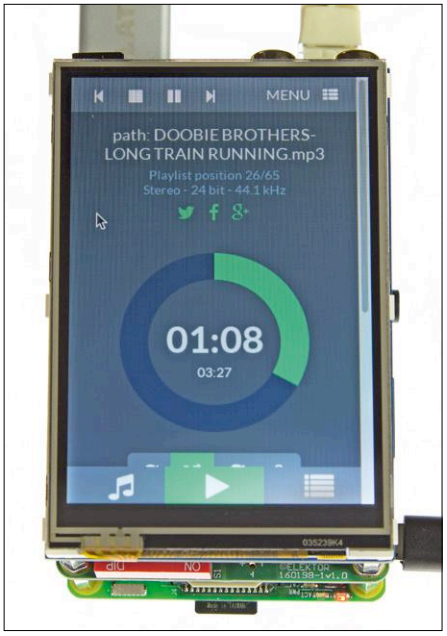


Bild 7. Volumio kann wahlweise über den Touchscreen (links) oder einen PC (rechts) bedient werden.



Weblinks

[1] Datenblatt PCM1794A: www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=pcm1794a&fileType=pdf

[2] Projektseite auf Elektor /Labs: www.elektormagazine.de/labs/audio-dac-for-rpi-networked-audio-player-using-volumio

IN DE STORE

→ **160198-1:**
Leerplatine

→ **160198-91:**
Aufgebautes Modul einschließlich Touchscreen (Kontaktleisten werden lose mitgeliefert)

→ **SKU 17631:**
Raspberry Pi 3 (Modell B)

Bei diesen Typen ist der Anschlussabstand geringer als bei den Standard-Ausführungen, diese passen hier ohne Weiteres nicht!
Das 3,5“-RPi-Touchscreen-Display von Waveshare muss 16 mm Abstand zur DAC-Platine einhalten. Wir haben auch hier vier metallene Abstandsbolzen mit M2,5-Schraubgewinde und Mutter verwendet, diesmal mit der Länge 14 mm. Die Kontaktleiste darf nicht bis zum Anschlag nach unten gedrückt werden! Im Zweifelsfall kann auch eine zweite Stapelkontaktleiste montiert werden, das Display sitzt dann etwas höher. Zwischen der 26-poligen Kontaktleiste des Displays und der DAC-Platine beträgt der Abstand ungefähr 2 mm. Die Bilder 5 und 6 zeigen den Netzwerk-Audiospieler nach Endmontage.

Messungen

Das Elektor-Labor hat den Audio-DAC messtechnisch untersucht, die wichtigsten Ergebnisse sind in den Tabellen 2 und 3 zusammengefasst. Bei den Messungen war S1 eingestellt wie in Tabelle 1 angegeben.
Die Stromaufnahme steigt mit der Samplefrequenz, Messwerte in Tabelle 2. Messungen der Totalen Harmonischen Verzerrungen plus Rauschen (THD+N) und der Intermodulationsverzerrungen (IMD) ergaben die Werte in Tabelle 3.
Auf der Projektseite zum Audio-DAC [2] haben wir ergänzende Messdiagramme veröffentlicht, die mit unserem Audio Precision Analyser aufgenommen wurden. ⏮

(160198)gd

Tipps und Tricks

Von Lesern für Leser

Hier kommt wieder eine clevere Lösung, die das Elektronikerleben leichter macht.

Digitalkamera als Testbildgenerator

Von Alfred Rosenkränzer (alfred_rosenkraenzer@gmx.de)

Zum Einmessen einer Videoüberwachungsanlage hätte ich vor einiger Zeit einen kleinen, mobilen und batteriebetriebenen Video-Testbildgenerator gebraucht. Aus meiner Zeit als professioneller Entwickler für Videotechnik besitze ich zwar diverse Generatoren, doch die sind weder klein noch batteriebetrieben. Also begann ich zu überlegen, was mit einer Kombination aus CPLD, RAM und DAC heute machbar wäre.

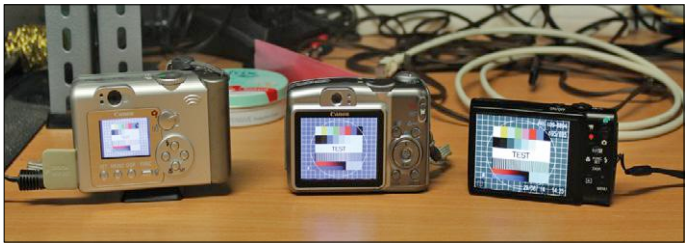


Bild 1. Das Testbild auf meinen drei Kameras.

Bis mir schlagartig auffiel, dass ich (und Sie vermutlich auch) so etwas ja bereits haben: All das steckt nämlich in einer Digitalkamera. Sie kann nicht nur die aufgenommenen Bilder auf ihrem eigenen Display anzeigen, sondern auch über eine Buchse auf einem externen Monitor. Ältere Kameras haben „nur" einen analogen Ausgang (oft via 3,5-mm-Klinkenbuchse), neuere sogar ein HDMI-Interface.

Fertige Testbilder gibt es zuhauf im Internet, zum Beispiel bei der Firma Burosch [1]. Doch leider wollte keine meiner Kameras ein Bild anzeigen, das sie (oder eine ähnliche Kamera) nicht selbst aufgenommen hatte. Den Versuch, in den Daten der EXIF-Datei die richtige Stelle zu finden und zu modifizieren, habe ich nach einiger Zeit aufgegeben. Falls ein Leser sich hier auskennt, wäre ich für entsprechende Hilfe dankbar.

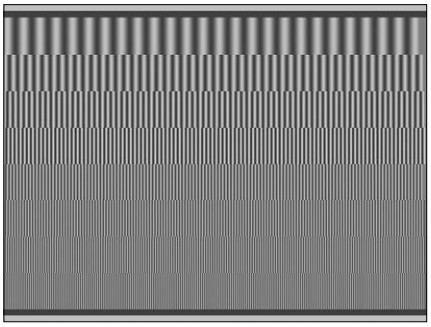


Bild 2. Das selbstgestrickte Multiburst-Testbild.

Um dieses Problem zu umgehen, machte ich mit der Kamera einfach ein Foto und tauschte den Bildinhalt mit einem Bildbearbeitungsprogramm aus. Für den analogen Ausgang genügt eine kleine Auflösung von 640 x 480 oder 800 x 600 Pixeln – mehr wird sowieso nicht dargestellt. Die Videonorm für PAL ist ja bekanntlich 720 x 576 und für NTSC 720 x 480 Pixel.

Bild 1 *zeigt das so gewonnene Testbild auf meinen drei Kameras. Um den Frequenzgang einer Übertragungstrecke zu messen (bei mir bis zu 80 m) kann man ein spezielles Video-Testsignal einsetzen, das Multiburst genannt wird. Wie der Name schon sagt, besteht es aus mehreren Bursts, beginnend mit niedriger Frequenz bis hin zur oberen Grenzfrequenz des Systems (ca. 5 MHz bei PAL und 4,2 MHz bei NTSC). Ein Multiburst war auch im genormten Testbild integriert, das früher in den TV-Sendepausen ausgestrahlt wurde. Am Ende der Übertragungsstrecke schaut man sich das Testbild mit einem Monitor oder das Signal mit einem Oszilloskop an. Haben alle Frequenzbursts die gleiche Amplitude, so ist der Frequenzgang flach und die Bandbreite groß. Die Bursts kann man in einer Zeile darstellen oder über das Bild von oben nach unten verteilen. Entsprechend spricht man von einem horizontalen oder vertikalen Multiburst.*

Da ich ein solches Testbild nicht finden konnte, habe ich mir selbst eines erzeugt (interessierte Leser können sich gerne per E-Mail an mich wenden, um Details zu erfahren). Am oberen und unteren Rand wurden die maximalen (hell) und minimalen Pegel (dunkel) als Referenzstreifen eingefügt. Der Anfang und das Ende jeder Zeile wird sanft ein- und ausgeblendet, um störende Überschwinger zu vermeiden. Das Ergebnis ist in **Bild 2** *zu bewundern.*

Die D/A-Wandler und das folgende Filter einer Kamera haben leider nicht die Qualität wie bei einem professionellen Testbildgenerator. Auch die Samplefrequenz ist nicht immer groß genug, um das 5-MHz-Spektrum auszuschöpfen. Man muss also Einschränkungen bei der maximalen Signalfrequenz und Bildqualität hinnehmen. Die von mir verwendeten Bild-Dateien kann man auf der Elektor-Webseite zu diesem Beitrag [2] downloaden. Falls Ihre Kamera sich so wiederborstig zeigt wie meine, können Sie wie oben erwähnt den Inhalt eines selbstgeschossenen Bilds mit Photoshop und Co. gegen so ein Testbild austauschen. ⏮

(160426)

[1] https://goo.gl/WHE0H9

[2] www.elektormagazine.de/160426

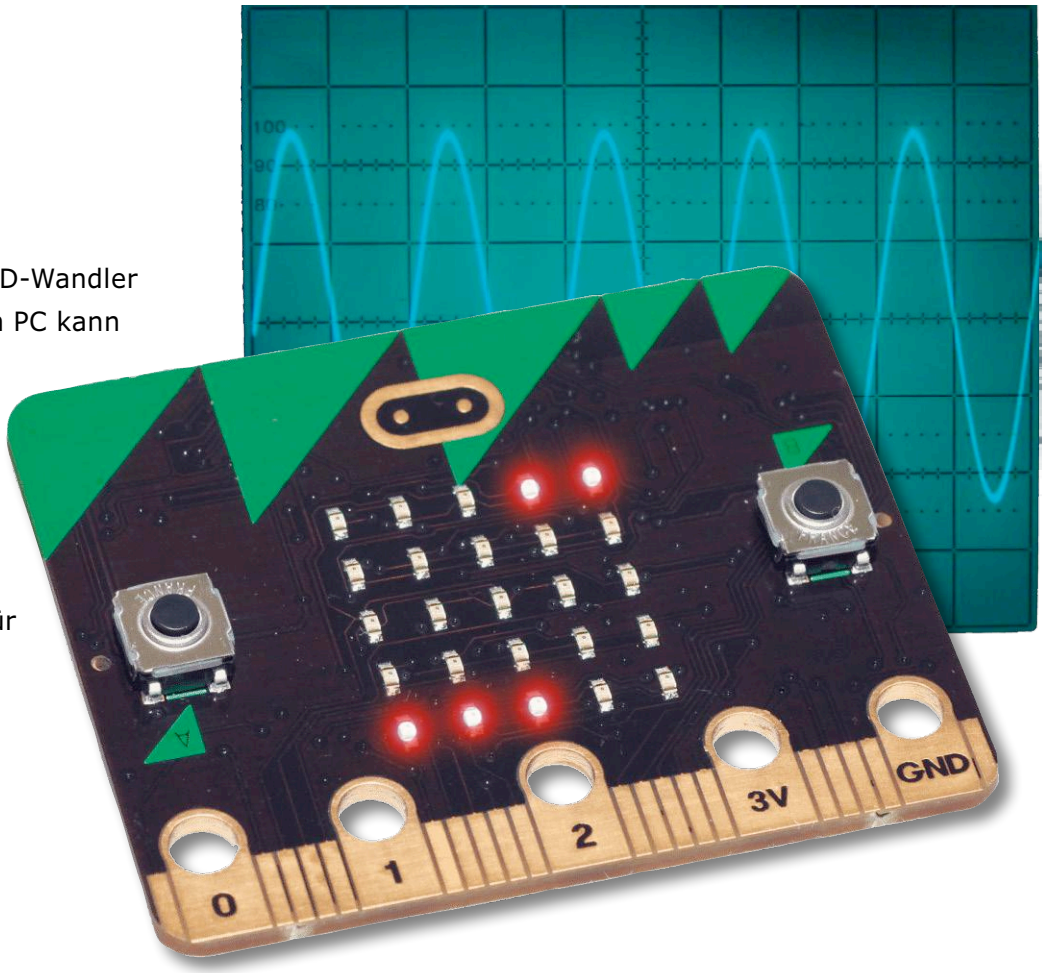
Sie haben selbst eine clevere Lösung für etwas wirklich Fummeliges? Wenden ein Bauteil oder Werkzeug auf ungewöhnliche Weise an? Haben eine Idee, wie man ein Problem einfacher oder besser angehen könnte, als das bisher gelöst wurde? Schreiben Sie uns – für jeden Tipp, den wir veröffentlichen, loben wir 40 Euro aus!

BBC micro:bit für Elektroniker (2)

Datenlogger und Oszilloskop

Von **Burkhard Kainka**

Jeder Mikrocontroller mit AD-Wandler und einer Schnittstelle zum PC kann zum Messkopf eines Datenerfassungssystems werden. Aber beim BBC micro:bit kommen noch ein kleines LED-Display und eine drahtlose Schnittstelle hinzu. Ideal für spezielle Anwendungen im Elektronik-Labor!



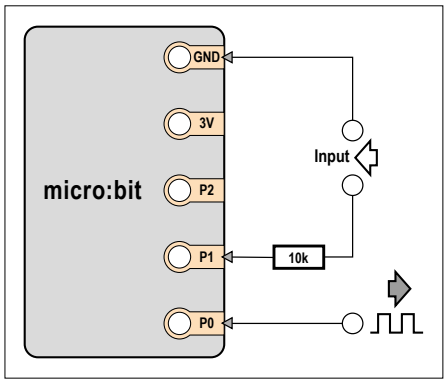


Bild 1. Messeingang und Signalausgang.

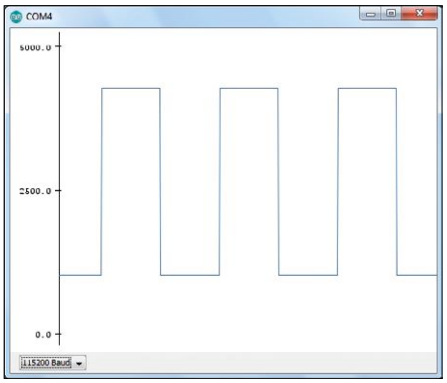


Bild 2. Ein Rechtecksignal mit 10 Hz.

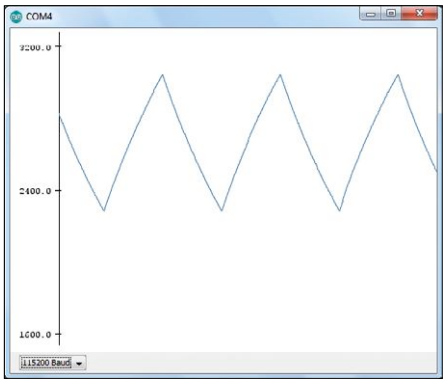


Bild 3. Das gefilterte Signal.

Gerade wegen der geringen Ausmaße des Micro:bit ist dieser für Aufgaben im Bereich der Messtechnik prädestiniert. Ob er über ein USB-Kabel oder per Batterie versorgt wird, über Bluetooth oder ein vereinfachtes Funk-Protokoll mit der Außenwelt kommuniziert, immer lässt er sich nahe beim eigentlichen Messobjekt unterbringen.

Ein USB-Oszilloskop

Für die Programmierung des Micro:bit hat sich die mbed-Plattform bewährt. Die ersten Schritte erklärt [1]. Alle Programme zu diesem Artikel stehen als Textfiles zum Download auf der Elektor-Seite bereit [2] und müssen in ein bestehendes mbed-Projekt kopiert werden.

Beim Einsatz des Micro:bit für allgemeine Messzwecke muss man sorgfältig darauf achten, den Messbereich zwischen GND und VCC nicht zu überschreiten. Für alle Fälle bewährt sich ein Schutzwiderstand von 10 kΩ in Reihe zum Eingang, der bei einer versehentlichen Überschreitung für die nötige Strombegrenzung sorgt (**Bild 1**). Außer dem analogen Eingang gibt es einen Rechteckausgang, mit dem man ein nützliches Testsignal erzeugen kann.

Das Programm nach **Listing 1** misst mit maximaler Geschwindigkeit ohne Zwischenspeicherung und mit sofortiger Übertragung jedes Messwerts. Ein wesentlicher Teil der Zykluszeit stammt dabei von der seriellen Übertragung mit 115200 Baud. Wenn dann einige Messwerte einstellig sind (3 mV) und andere vierstellig (3000 mV), hat man keinen gleichmäßigen Ablauf der Messung. Deshalb wird hier die Spannung um 1000 mV hochgesetzt, womit die möglichen Messwerte von 1000 mV bis 4300 mV reichen und immer die gleiche Zeit benötigen. Das Programm bringt gleich auch eine Signalquelle mit, damit man ohne zusätzlichen Aufwand etwas zu messen findet. P1 wird zu einem PWM-Ausgang mit einer PWM-Frequenz von 10 Hz und einer Periode von 100 ms.

Für die Auswertung gibt es viele Möglichkeiten. Man könnte die Daten mit einem Terminalprogramm empfangen und dann mit einer Tabellenkalkulation darstellen. Eine bequeme Alternative ist der Serielle Plotter aus der Arduino-IDE ab Version 1.6.8. Die Software bietet einen durchlaufenden Bildschirm und eine automatische Anpassung des Anzeigebereichs, sodass man auf eine Bereichsumschaltung der Anzeige verzichten kann. Mit den gegebenen Einstellungen misst man ein symmetrisches Rechtecksignal mit 10 Hz (**Bild 2**). Damit hat man zugleich eine bekannte Zeitachse. Das gesamte Oszillogramm umfasst offensichtlich eine Messzeit von 300 ms.

Der Serielle Plotter der Arduino-IDE plottet immer zuerst von links nach rechts, bis der Schirm gefüllt ist. Danach läuft das Bild nach links durch, sodass alte Daten verschwinden. Das Problem der durchlaufenden oder stehenden Darstellung wird mit dem Taster A gelöst (`if(uBit.buttonA.isPressed()` im Listing 1). Man drückt für die Dauer der

Listing 1. Schnelle Messung mit direkter Datenübertragung.

```
//Voltage Logger/Scope
#include "MicroBit.h"
MicroBit uBit;

int main()
{
   uBit.init();
   MicroBitSerial serial(USBTX, USBRX);
   uBit.io.P1.setAnalogValue(512);
   uBit.io.P1.setAnalogPeriodUs(100000);
   while (1) {
     if(uBit.buttonA.isPressed()){
        int u = 1000+3300 * uBit.io.P0.getAnalogValue()/ 1023;
        uBit.serial.printf("%d\r\n", u);
       // uBit.sleep(100);
        }
      }
}
```

Listing 2. Schnelles Speichern und anschließende Übertragung.

```
//Fast Scope
#include "MicroBit.h"
MicroBit uBit;

int main(){
  char d[400];
   uBit.init();
   MicroBitSerial serial(USBTX, USBRX);
   uBit.io.P1.setAnalogValue(512);
   uBit.io.P1.setAnalogPeriodUs(2000);
   while (1) {
     if(uBit.buttonA.isPressed()){
         for(int i = 0; i < 400; i++){
              d[i]  =  uBit.io.P0.getAnalogValue()/ 4;
              }
        for(int i = 0; i < 50; i++) uBit.serial.printf("%d\r\n", 0);
         for(int i = 0; i < 400; i++){
              uBit.serial.printf("%d\r\n", d[i]);
              }
           for(int i = 0; i < 50; i++) uBit.serial.printf("%d\r\n", 255);
      }
      uBit.sleep(500);
   }
}
```

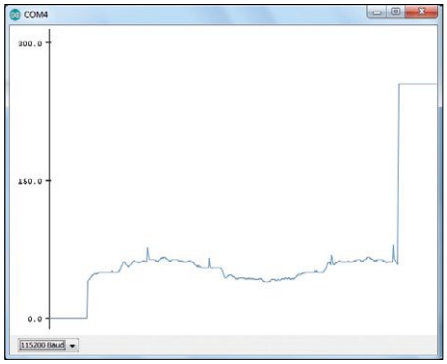


Bild 4. Messung mit höherer Abtastrate.

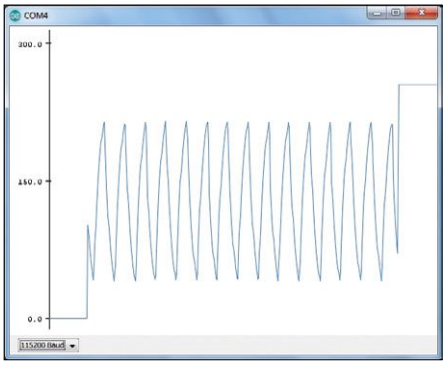


Bild 5. Ein Sägezahnsignal mit 500 Hz.

Messung auf den Knopf A und erhält eine durchlaufende Anzeige. Sobald man loslässt, bleibt die letzte Anzeige stehen, sodass man sie genauer betrachten oder eine Kopie davon speichern kann.
Der zusätzliche Signalausgang kann nützlich werden, wenn man Schaltungen oder Bauteile untersuchen will. Mit einem Tiefpassfilter mit 4,7 k und 22 µF entsteht aus dem Rechteck wie erwartet ein Sägezahnsignal (**Bild 3**). Der Messbereich des Arduino-Plotters passt sich automatisch an die kleineren Spannungen an.

Erhöhung der Abtastrate durch Zwischenspeicherung

Wenn man die serielle Übertragungszeit einsparen kann, bestimmt allein die Messzeit des AD-Wandlers die erreichbare Abtastrate. Man bildet also ein Daten-Array, füllt es mit Messdaten und sendet diese anschließend an den PC. Aber da fangen die Probleme erst an, mit denen man vielleicht nicht gerechnet hätte. Obwohl der Controller reichlich RAM besitzt, kommt bereits ein Array vom Typ `int d[100]` an die Grenzen, weil die Microbit-Runtime nicht mehr viel übrig lässt. Wenn man aber den Seriellen Plotter nutzen will, sollten es schon 500 Messwerte sein.
Allerdings muss man wissen, dass der Typ `int` in einem 32-Bit-System eine Breite von vier Bytes hat. Es stehen also 400 Bytes zur Verfügung. Deshalb wird nun mit `char d[400]` ein Byte-Array mit 400 Bytes verwendet. Durch eine Division durch 4 werden aus den 10-Bit-Daten des AD-Wandlers sparsame 8 Bit.

Es werden also nun 400 Bytes gespeichert und übertragen (**Listing 2**). Da fehlen eigentlich noch 100 Bytes, um den Plotter zu füllen. Aber man kann aus der Not eine Tugend machen und einen Vorspann mit 0-Bytes vorwegschicken und einen Nachspann mit 255-Bytes dranhängen. Das veranlasst den seriellen Potter dann, immer den vollen Messbereich darzustellen und gibt dem Betrachter eine klare Orientierung bezüglich der Bereichsgrenzen.
Das Messergebnis in **Bild 4** zeigt ein 50-Hz-Signal mit den typischen Störimpulsen, die man mit einem offen liegenden Messkabel einfängt. Da etwa 1,5 Schwingungen gezeigt werden, dauert die Messung rund 30 ms, was bei 400 Messpunkten auf eine Abtastrate von etwa 13 kHz schließen lässt. Weil die PWM-Frequenz in diesem Programm auf 500 Hz hochgesetzt wurde, kann man dies leicht mit dem eigenen Signal verifizieren. **Bild 5** zeigt das PWM-Signal am Ausgang eines Tiefpassfilters mit 4,7 kΩ und 100 nF.

Drahtlose Übertragung von Messdaten

Der BBC micro:bit ist mit Bluetooth Low Energy (BLE) ausgestattet. Einen besonderen Chip dafür sucht man auf der Platine vergeblich, denn die HF-Baugruppen sind bereits im Mikrocontroller enthalten. Der nRF51822 von Nordic Semiconductor wurde ursprünglich für Anwendungen wie drahtlose Tastaturen und Mäuse entwickelt, die keine große Reichweite benötigen, aber sparsam mit der Energie einer Batterie umgehen sollten. Davon profitiert auch der Micro:bit. Man kann die Platine mit einer 3-V-Batterie versorgen und dann sogar auf das USB-Kabel verzichten. Damit ist das System auch für batteriebetriebene Langzeitanwendungen tauglich. Die wichtigsten Kurzdaten des nRF51822 sprechen für sich:

- 2.4 GHz transceiver
- –93 dBm sensitivity in Bluetooth® low energy mode
- 250 kbps, 1 Mbps, 2 Mbps supported data rates
- TX Power -20 to +4 dBm in 4 dB steps

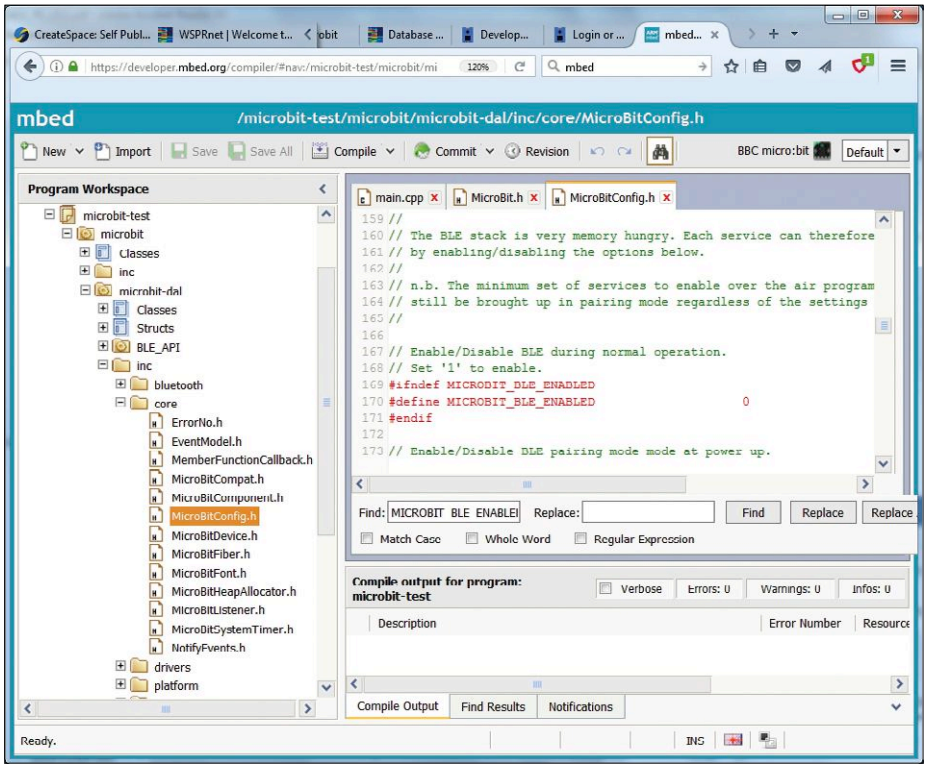


Bild 6. Deaktivieren von BLE.

- TX Power -30 dBm Whisper mode
- 13 mA peak RX, 10.5 mA peak TX (0 dBm)
- 9.7 mA peak RX, 8 mA peak TX (0 dBm) with DC/DC
- RSSI (1 dB resolution)
- ARM® Cortex™-M0 32 bit processor
- 275 µA/MHz running from flash memory
- 150 µA/MHz running from RAM
- Serial Wire Debug (SWD)

Die Programmierung mit mbed erlaubt eine Verwendung von Bluetooth, sodass man Daten direkt an ein Smartphone oder Tablet senden könnte. Allerdings ist dafür ein recht großer Software-Stack erforderlich, und auch auf der Gegenseite müsste man noch die geeignete App entwickeln.

Aber es geht auch einfacher. Man kann nämlich den 2,4-GHz-Transveiver auf einer unteren Ebene ansprechen und auf das komplizierte Bluetooth-Protokoll verzichten. Dafür gibt es die MicroBitRadio-Unterstützung mit einfachen Datagrammen, also Kurznachrichten, die den Transceiver direkt verwenden. Ein Micro:bit sendet eine Kurznachricht, alle anderen in Reichweite empfangen sie. Dazu wurden ein Default-Kanal und eine Default-Sendeleistung festgelegt, sodass man sich eigentlich um nichts kümmern muss. Es ist damit eine der einfachsten Methoden, drahtlos Daten zu übertragen. So hat man zugleich auch eine Möglichkeit, potentialfrei zu messen. Eine typische Anwendung könnte ein EKG sein, weil man mit einer Potentialtrennung die lästigen Brummstörungen loswird.
Alles was man braucht sind zwei Micro:bit. Einer wird als Messgerät verwendet, das Daten sendet, ein zweiter wird als Empfänger programmiert, der die Daten anzeigt oder über USB an einen PC übermittelt. Um das MicroBitRadio in mbed zu verwenden muss man allerdings Bluetooth Low Energy deaktivieren. Dazu gibt es einen Hinweis in der BBC micro:bit Dokumentation:

„It is not currently possible to run the MicroBitRadio component and Bluetooth Low Energy (BLE) at the same time. If you want to use the MicroBitRadio functionality, you need to disable the BLE stack on your micro:bit by compiling the runtime with #define MICROBIT_BLE_ENABLED 0 in your inc/MicroBitConfig.h file.“

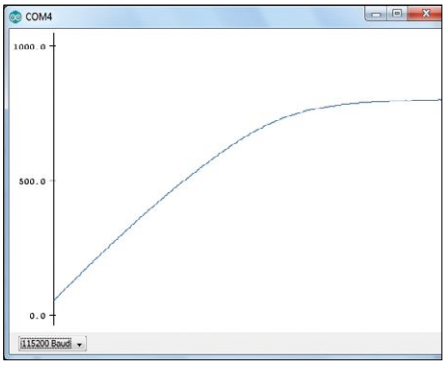


Bild 7. Über Funk gesendete Messdaten.

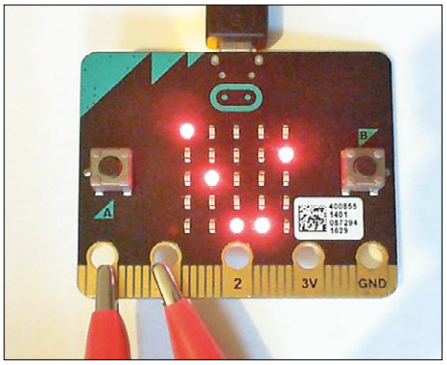


Bild 8. Messung des eigenen Testsignals.

Es ist nicht ganz einfach, die richtige Stelle in den vielen Dateien des Runtime-Systems zu finden. Der genaue Pfad lautet: *microbit\microbit-dal\inc\core\MicroBitConfig.h* (siehe **Bild 6**). In dieser Datei gibt es einen Eintrag `MICROBIT_BLE_ENABLED 1`, in dem man die 1 durch eine 0 ersetzen muss. Beim nächsten Kompilieren ist dann BLE deaktiviert, sodass man das vereinfachte MicroBitRadio verwenden kann.

Das Programm in **Listing 3** sendet und empfängt Datagramme mit Messdaten. Sender und Empfänger können deshalb dasselbe Programm verwenden und auch gegenseitig Daten austauschen. Der Sender führt eine Messung an P1 durch und sendet das Ergebnis in mV. Der Empfänger reicht die empfangenen Daten über den USB weiter. Auf der PC-Seite ändert sich nichts. Hier kann wieder der Serielle Plotter eingesetzt werden. **Bild 7** zeigt

Anzeige

ein Messergebnis. Der Sender wurde per Batterie betrieben und befand sich in drei Metern Entfernung vom Empfänger. Bis etwa 10 m funktioniert die Verbindung. Am analogen Eingang P1 war ein Elko mit 10 µF angeschlossen. Er wurde langsam über den auf der Platine vorhandenen Pullup-Widerstand von 10 MΩ geladen. Man erkennt Abweichungen von der üblichen Ladekurve, weil dieser Elko lange nicht im Einsatz war. In so einem Fall fließt zuerst ein kleiner Leckstrom, der nur allmählich abnimmt. Im rechten Teil des Diagramms liegt die gemessene Spannung immer noch deutlich unter 1 V und steigt nur langsam an.

Mini-Oszilloskop mit LED-Anzeige

Ein ganz einfaches Oszilloskop ist besser als keins, und manchmal kommt es mehr darauf an, dass das Gerät sehr klein, autonom und handlich ist. Hier werden die Messdaten grafisch auf dem LED-Display mit 5 mal 5 LEDs dargestellt (**Listing 4**). Auch wenn man durch ein sehr viel besseres Gerät verwöhnt ist, kann man durchaus damit arbeiten. Es ist ganz erstaunlich, was mit einem so einfachen Oszilloskop noch erkennbar ist.

Das Mini-Oszilloskop verwendet wieder den Port 1 als analogen Eingang und nutzt zusätzlich den Port 0 als PWM-Ausgang. Bei einer Periode von 500 µs entsteht ein Ausgangssignal mit einer Frequenz von 2 kHz. Eine direkte Verbindung zum Messeingang zeigt die Grenzen des AD-Wandlers (**Bild 8**). Die Abtastzeit ist offensichtlich zu lang, um scharfe Flanken des PWM-Signals abzubilden. Die Grenzfrequenz des einfachen Oszilloskops liegt also irgendwo unter 10 kHz. Das reicht zwar noch nicht so ganz für das HF-Labor, wohl aber für viele einfache Messungen und Experimente. ◀

(160384)

Listing 3. Senden und Empfangen von Datagrammen.

```
//Radio Data
#include "MicroBit.h"
MicroBit uBit;


void onData(MicroBitEvent e)
{
    ManagedString s = uBit.radio.datagram.recv();
    uBit.serial.send (s);
    uBit.serial.send (" \r\n");
}

int main()
{
   uBit.init();
   uBit.messageBus.listen(MICROBIT_ID_RADIO, MICROBIT_RADIO_EVT_
        DATAGRAM, onData);
   uBit.radio.enable();
   char output[16];
   while (1) {
        int u = 3300 * uBit.io.P1.getAnalogValue()/ 1023;
        itoa (u, output);
        uBit.radio.datagram.send(output);
        uBit.sleep(100);
    }
}
```

Listing 4. Nutzung der LED-Anzeige.

```
//LED-Scope
#include "MicroBit.h"
MicroBit uBit;
int main()
{
   int y;
   uBit.init();
   uBit.io.P0.setAnalogValue(512);
   uBit.io.P0.setAnalogPeriodUs(500);
   uBit.display.enable();
   MicroBitImage image(5,5);
   while (1) {
        for(int x = 0; x < 5; x++){
            y =  4- (uBit.io.P1.getAnalogValue()/205);
            image.setPixelValue(x,y,255);
        }
        uBit.display.print(image);
        uBit.sleep(500);
        image.clear();
    }
}
```

Weblinks

[1] www.elektormagazine.de/160273

[2] www.elektormagazine.de/160384

[3] https://developer.mbed.org/

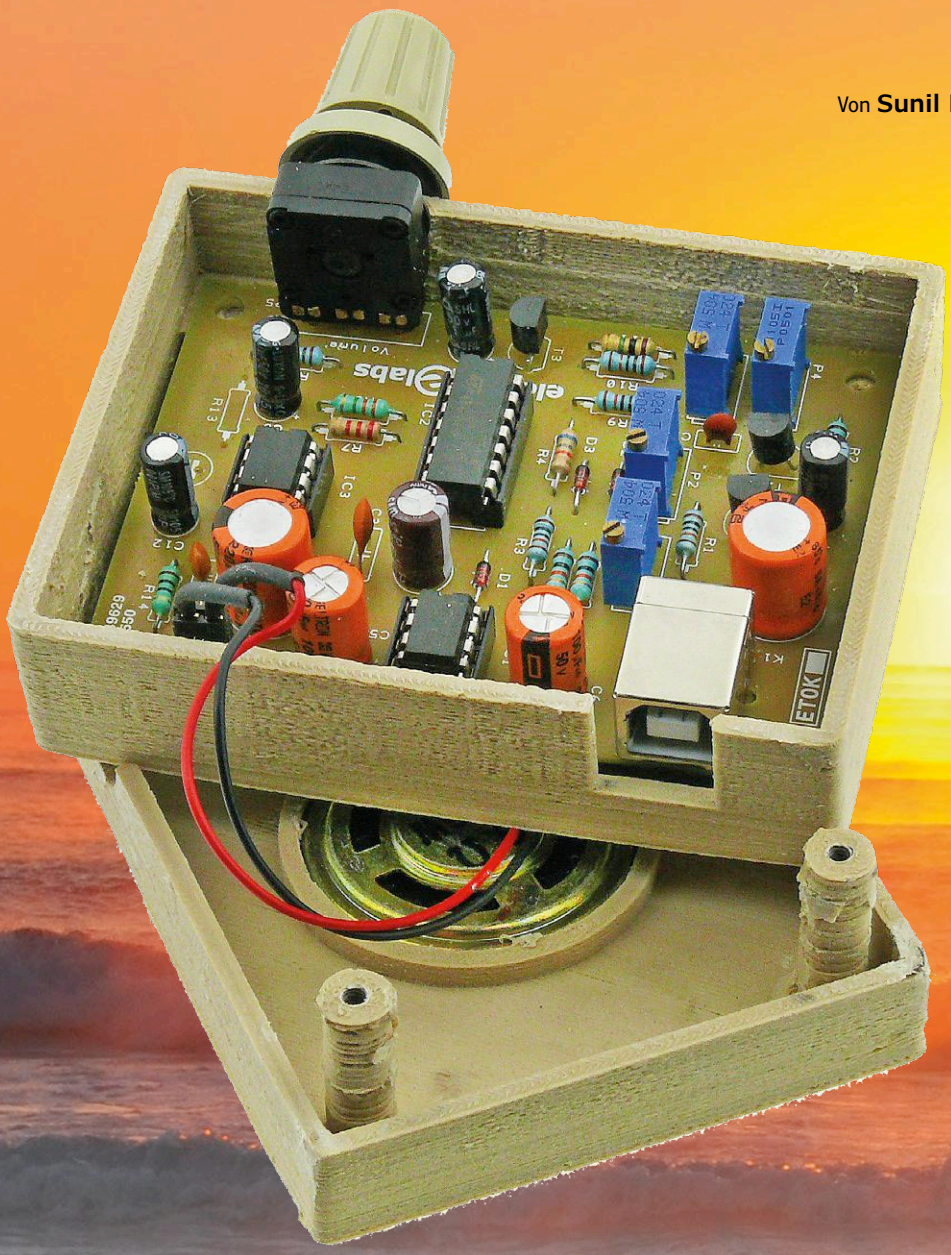
[4] https://lancaster-university.github.io/microbit-docs/ubit

[5] B. Kainka, Micro:bit Praktikum, CreateSpace 2016

Meeresrauschen-Simulator

Sanft einschlafen im Rauschen der Wellen

Von **Sunil Malekar** und **Clemens Valens** (Elektor-Labor)



Wem Hypnos als Gott des Schlafes gerade nicht gut gesonnen ist, der kann Poseidon darum bitten, ihn mit den Geräuschen seiner Meere zu entspannen und in den Schlaf zu lullen. Die folgende Schaltung wird das Sandmännchen ganz bestimmt zu ihrem Bett locken.

Man könnte annehmen, dass es für eine Zeitschrift wie *Elektor* ein paar Elektronik-Ingenieure plus Labor für Entwicklungen, ein paar Redakteure für Text und Recherche sowie Grafiker für passende Illustrationen braucht. Doch weit gefehlt. Damit Sie regelmäßig Elektor lesen können, braucht es jemand mit dem Talent, das alles zusammen- und im Fluss zu halten. Bei Elektor ist das unsere kaum zu unterschätzende Power-Planerin Hedwig. Auch wenn Elektronik nicht ihr Spezialgebiet ist (Originalzitat: *„Ist das so ein Kondensator-Ding oder wie nennt man das kleine blaue Teil schon wieder?“*), so mag Hedwig

INFOS ZUM PROJEKT

Rauschen
Simulator
Entspannung | Meer

➡ Einsteiger
Fortgeschrittener
Experte

Etwa 1 Stunde

Lötkolben,
kleiner Schraubendreher,
Bett

Rund 15 €

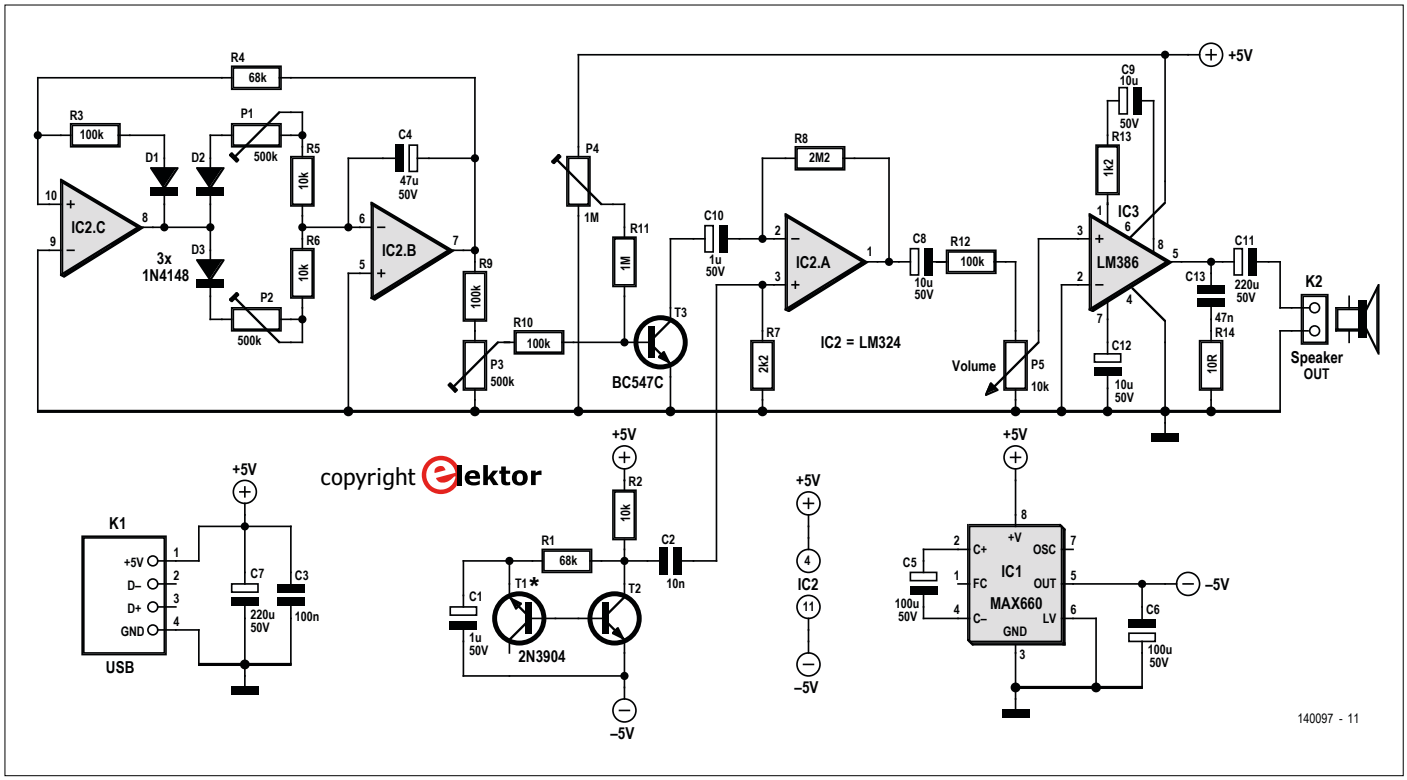
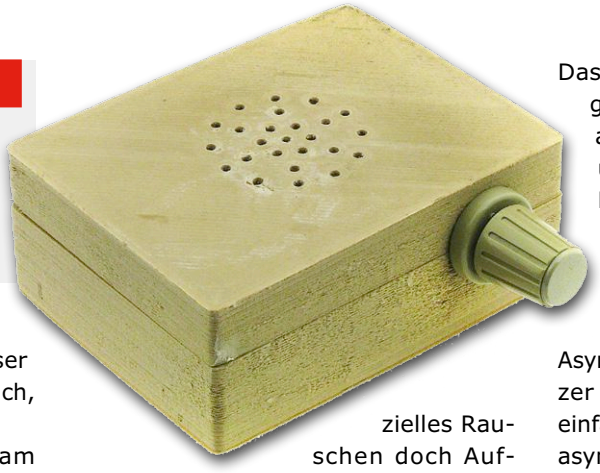


Bild 1. Elektronik simuliert Brandung.

Eigenschaften

- Anstieg und Abfall der Wellen unabhängig einstellbar
- Betrieb mit einem kleinen 5-V-Netzteil
- Kein (0) Arduino inside ;-)



doch ganz spezielle Elektronik: Dieser Meeresrauschen-Generator ist für Dich, Hedwig!

Der Sound von Wellen, die sich am Strand brechen, wirkt auf viele Zeitgenossen beruhigend. So entspannend oder gar romantisch das sein mag – für einen unparteiischen und unsentimentalen Wissenschaftler handelt es sich bei diesen Geräuschen lediglich um modulierte Rauschsignale. Klingt vielleicht wie rosa Rauschen, aber trotz der „Farbe" ist es eben nichts als Rauschen. Und das kann man elektronisch einfach erzeugen. Faktisch ist es sogar viel schwieriger zu erreichen, dass eine Schaltung nicht rauscht ;-).

Für ein Rauschsignal benötigt man eigentlich nicht mehr als eine in Sperrrichtung betriebene Diode.

Die Schaltung

Auch wenn das Erzeugen von Rauschen im Prinzip einfach ist, so erfordert spezielles Rauschen doch Aufwand. **Bild 1** zeigt die Schaltung unseres Projekts. T1 als Diode in Sperrrichtung ist die Rauschquelle. Der Emitter fungiert hier als Kathode und die Basis als Anode. Der durch diese Pseudo-Diode fließende Rauschstrom wird durch T2 gepuffert und verstärkt. Schaut man genau hin, so fällt auf, dass diese beiden Transistoren zwischen +5 V und -5 V liegen. Eine Spannung von 10 V ist nötig, damit die BE-Strecke von T1 als Pseudo-Z-Diode in den Durchbruchsbereich kommt. T1 und T2 sind vom Typ 2N3904, denn es zeigte sich, dass diese Fabrikate mehr rauschen als die bei Elektor gebräuchlicheren Typen (BC547 u.a.). Wenn Sie mit verschiedenen Typen experimentieren wollen: Die für die Platine vorgesehenen 2N3904 haben eine andere Pin-Belegung als BC547 & Co.

Das erzeugte Rauschen ist noch nicht genug für die Simulation von Wellen am Strand. Es braucht Modulation und diese wiederum einen Oszillator. Eine Hüllkurve wie etwa eine asymmetrische Sinuswelle scheint den realistischsten Effekt zu ergeben, was die Sache allerdings etwas komplizierter macht. Besonders die Asymmetrie (das Anschwellen sollte kürzer dauern als der Abfall) ist nicht so einfach zu verwirklichen. Der Bau eines asymmetrischen Dreiecksgenerators ist aber einfach – daher die Schaltung rund um IC2.C und IC2.B. Das Ergebnis ist sehr gut brauchbar. Normalerweise produziert so eine Schaltung ein Dreiecksignal, das symmetrisch um die Nulllinie liegt. Das brauchen wir hier aber nicht, weshalb der Modulator T3 schlicht die negative Halbwelle abblockt. Dies aber ergäbe diskontinuierliche Unterbrechungen. Damit genau das nicht passiert, wurde D1 hinzugefügt. Jetzt schwankt der Ausgang von IC2.B nett zwischen 0 V und etwa 3 V.

Mit den Potis P1 und P2 stellt man die modulierende Hüllkurve ein. P1 beeinflusst das Ansteigen und P2 das Abfallen des Pegels. Additiv bestimmen sie die Frequenz der Modulation bzw. den Wellenschlag. Je kleiner der Gesamtwert, desto schneller kommen die Wellen. Ihr

Verhältnis bestimmt die Symmetrie der Hüllkurve.
T3 dient als stromgesteuerter Widerstand und moduliert so die Verstärkung von IC2.A. Je höher der Basisstrom, desto geringer der Widerstand und desto höher der Pegel des Ausgangssignals. P3 beeinflusst quasi die Wellenhöhe und P4 das Grundrauschen.
Der Ausgang von IC2.A gelangt über den Lautstärkesteller P5 an den Audioverstärker IC3. Hier handelt es sich um die klassische LM386-Schaltung zum direkten Anschluss eines Lautsprechers oder Kopfhörers. Die Maximalverstärkung von IC3 wird durch R13 und C9 festgelegt. Ohne die beiden Bauteile beträgt sie Faktor 20, wie angegeben liegt sie bei etwa 50. Und ersetzt man R13 durch eine Drahtbrücke, kommt man immerhin auf 200.
Alles außer IC3 wird mit symmetrischen ±5 V betrieben. Trotzdem reicht ein normales 5-V-Steckernetzteil aus. IC1 ist nämlich ein Spannungsinverter mit geschaltetem Kondensator und sorgt so für die -5 V.

Kalibration

Wie bei jeder Präzisionselektronik gibt es auch hier Potis, die für optimale Ergebnisse justiert werden müssen. Doch das ist einfach: Zuerst wird P3 auf Minimum (Schleifer auf Masse) gestellt. Mit P4 stellt man nach Geschmack das Hintergrundrauschen ohne Wellen ein.
Nun fügt man mit P3 den gewünschten Wellenschlag (seine Amplitude) hinzu. Bei Begrenzung reduziert man P4 ein Stück. P3 und P4 beeinflussen sich gegenseitig, erfordern also Nachjustage.
Zum Schluss stellt man mit P1 und P2 die Form und Frequenz der Wellen ein. Unsere Tests haben ergeben, dass für einen realistischen Klang die Anstiegszeit (mit P1) etwa 5...10 Mal kürzer sein sollte als der Abfall (mit P2) – die Wellenform ähnelt daher eher einem Sägezahn als einem Dreieck. Als Periodendauer sind mehrere Sekunden adäquat. Einfach ausprobieren!

(140097)

EINKAUFSZETTEL

→ 140097-1
Platine

Weblink

[1] www.elektormagazine.de/140097

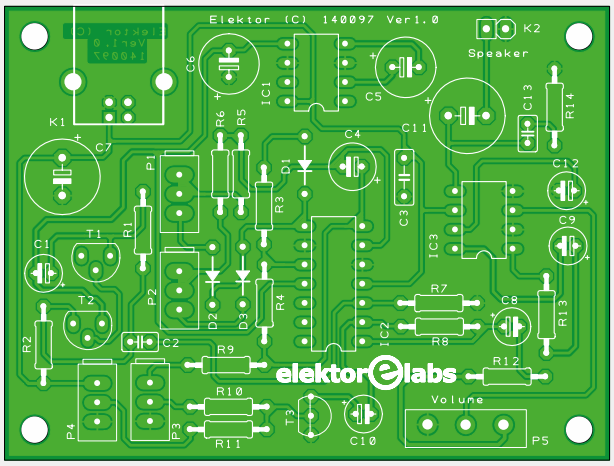
STÜCKLISTE

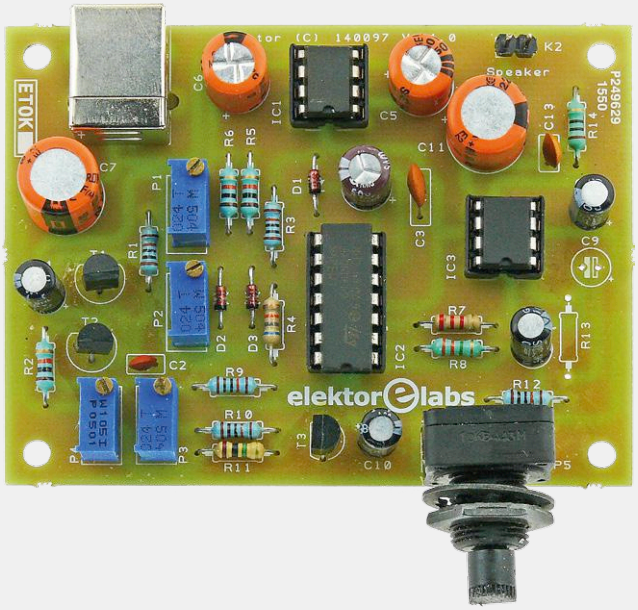
Widerstände:
R1, R4 = 68 k
R2, R5, R6 = 10 k
R3, R9, R10, R12 = 100 k
R7 = 2k2
R8 = 2M2
R11 = 1 M
R13 = 1k2
R14 = 10 Ω
P1..P3 = 500-k-Trimmpoti, klein, stehend
P4 = 1-M-Trimmpoti, klein, stehend
P5 = 10-k-Poti, logarithmisch

Kondensatoren:
C1, C10 = 1 µ / 50 V
C2 = 10 n
C3 = 100 n
C4 = 47 µ / 50 V
C5, C6 = 100 µ / 50 V
C7, C11 = 220 µ / 50 V
C8, C9, C12 = 10 µ / 50 V
C13 = 47 n

Halbleiter:
D1..D3 = 1N4148
IC1 = MAX660
IC2 = LM324
IC3 = LM386
T1, T2 = 2N3904
T3 = BC547C

Außerdem:
K1 = USB-B-Buchse
K2 = 2-pol. Stiftleiste, RM 0,1"
Platine # 140097-1



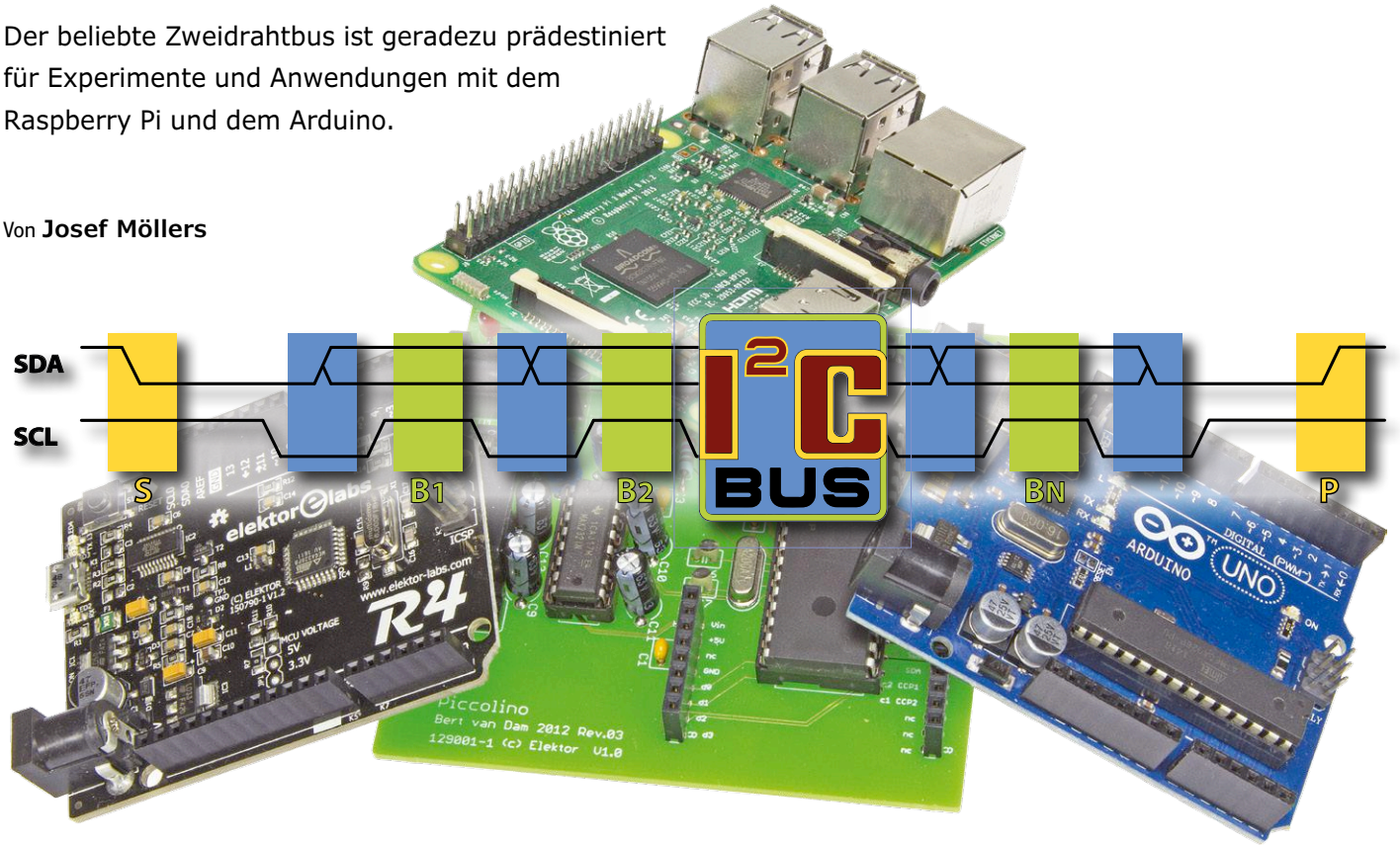


Der I²C-Bus

Teil 2: Der Bus am Controller

Der beliebte Zweidrahtbus ist geradezu prädestiniert für Experimente und Anwendungen mit dem Raspberry Pi und dem Arduino.

Von **Josef Möllers**



Raspberry Pi, BeagleBone, Arduino, Genuino, ATmega, PIC, fast jeder PC: Beinahe alles, was heute als Rechenknecht bei Makern auf dem Labortisch steht oder liegt, verfügt über einen (oder mehrere) I²C-Anschlüsse. Anhand eines I²C-Temperatursensors LM75 soll nun beschrieben werden, wie man die I²C-Interfaces von Raspberry Pi, ATmega und Arduino als Master und, wo möglich, auch als Slave einsetzen kann.

Raspberry Pi

Der Raspberry Pi verfügt physikalisch über zwei I²C-Busse, von denen aber in der Regel nur einer direkt nutzbar ist. Die Pull-Up-Widerstände nach 3,3 V sind beim Raspberry Pi bereits vorhanden und immer aktiv. SDA und SCL liegen auf Pin 3 (SDA) und Pin 5 (SCL) der Erweiterungs-Stiftleiste, praktischerweise ganz in der Nähe von 3,3 V (Pin 1), 5 V (Pin 2 und 4) und Masse (Pin 6)! Diese Pins gehören zum I²C-Bus 1. Mit einem kleinen Aufsteck-Platinchen ist also schnell ein Temperatursensor, eine Real-Time-Clock oder ein Lagesensor platzsparend installiert. **Achtung**: Die Ports des Raspberry Pi sind **ausschließlich mit 3,3 V** zu beschalten, ein Kontakt mit 5 V kann zur Zerstörung des Raspberry Pi führen! Deswegen sollte man eine Schaltung erst gut überprüfen, bevor man sie mit den Ports des Raspberry Pi verbindet. Insbesondere ist darauf zu achten, dass ein I²C-Slave selbst keine Pull-Up-Widerstände nach 5 V besitzt. Gegebenenfalls müssen diese entfernt werden, sie haben auf einem Slave ohnehin nichts verloren.

Aufgrund der komfortablen Umgebung eignet sich der Raspberry Pi gut dazu, sich mit neuen, unbekannten I²C-Slaves vertraut zu machen. **Bild 1** zeigt den Anschluss eines Experimentierboards mit einem LM75 am Raspberry Pi.

Master-Betrieb

Um den I²C-Bus des Raspberry Pi unter Raspbian benutzen zu können, muss man zuerst zwei zusätzliche Treiber laden. Dazu startet man `raspi-config` und wählt im Menüpunkt `Advanced Options` den Unterpunkt `I2C`, um die Schnittstelle einzuschalten und die Kernel-Module zu laden. Alternativ trägt man mit dem Editor in der Datei `/etc/modules` Folgendes ein:

```
i2c-dev
i2c-bcm2708
```

Nach einem Neustart sind die beiden Treiber (und eventuell erforderliche weitere Treiber) geladen und die Geräteknoten `/dev/i2c-<n>` existieren, was man mit folgenden Kommandos ($ …) bestätigt bekommt:

```
$ lsmod | fgrep i2c_
i2c_dev          XXXX  0
i2c_bcm2708      YYYY  0
$ ls /dev/i2c-*
dev/i2c-1
```

Hier stehen XXXX und YYYY für die Größe der Module, die beiden Nullen deuten an, dass noch keiner diese Module benutzt. Die Kommandos funktionieren übrigens auch ohne Root-Berechtigung.
Zusätzlich installiert man das Paket `i2c-tools`, das unter anderem einen Detektor für I2C-Bausteine und –Busse enthält:

```
sudo apt-get install i2c-tools
```

Für einige I²C-Peripheriebausteine bringt Raspbian bereits Treiber mit, so etwa für die Real-Time-Clock (RTC) RV-8523. Ohne weitere Einstellungen betreibt der Raspberry Pi den I²C-Bus im Standard-Modus mit 100 kBps.
Zuerst einmal kann man sich im Terminal mit dem Tool `i2cdetect` einen Überblick über die vom System am I²C-Bus 1 erkannten Slaves verschaffen. Das Ergebnis sieht dann so aus wie in **Bild 2**, wenn sich der LM75 auf Adresse `0x48` meldet.
Mit den Befehlen `i2cget, i2cset` und `i2cdump` kann man ganz ohne eigene Programmierung mit dem LM75 kommunizieren (`0x00` ist dabei die Registernummer und muss immer angegeben werden):

```
pi@raspberrypi ~ $ i2cget -y 1 0x48 0x00 w
0xa010
```

Die beiden Bytes der Antwort müssen nun vertauscht werden `(0x10a0)`. Vom Ergebnis sind nur die oberen neun Bits gültig `(0x021)`. Der LM75 liefert die Temperatur in 0,5-K-Schritten, die Temperatur lag zu dem Zeitpunkt also bei 16,5 °C.
Die Hardware des Raspberry Pi kann theoretisch auch als I²C-Slave betrieben werden, jedoch unterstützt der Linux-Treiber dies nicht.

Programmierung in C und Python
Für die Programmierung des I²C-Busses **in C** werden fünf Funktionen benötigt:

- `open()` für den Zugriff auf den I²C-Geräteknoten,
- `ioctl()` für die Einstellung von Parametern des I²C-Slaves,
- `read()` und `write()` für die eigentliche Kommunikation mit dem Slave und
- `close()` für die Beendigung des Zugriffs auf den I²C-Geräteknoten.

Für die nachfolgend verwendete Definition `I2C_SLAVE` wird, zusätzlich zu den anderen include-Dateien, die Datei `linux/i2c-dev.h` benötigt:

```
# include <linux/i2c-dev.h>
```

Der Zugriff auf den Geräteknoten wird mit `open()` realisiert:

```
fd = open("/dev/i2c-1", O_RDWR);
```

Danach setzt man die Slave-Adresse mit einem Aufruf von `ioctl()`:

```
ioctl(fd, I2C_SLAVE, 0x48);
```

Nun kann man schreiben und lesen:

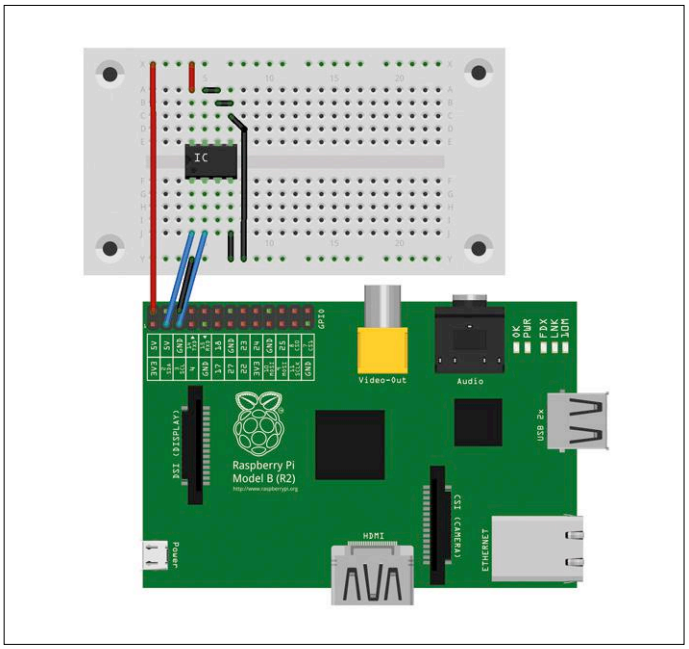


Bild 1. Anschluss des LM75 auf einem Experimentierboard an den Raspberry Pi.

```
unsigned char buf[2];
float T;
buf[0] = 0;
write(fd, buf, 1);   /* write register number 0 */
read(fd, buf, 2);    /* read temperature register */
T = ((buf[0]<< 8) | buf[1]) / 256.0;
```

Am Ende beendet man den Zugriff auf den Geräteknoten wieder:

```
close(fd);
```

Selbstverständlich sollte man die Rückgabewerte der Funktionsaufrufe besser auswerten, um Fehler festzustellen. Zum Beispiel um zu detektieren, ob man überhaupt die Berechtigung hat, `/dev/i2c-1` zu öffnen, ob der Slave mit der Adresse `0x48` existiert oder ob die Übertragung erfolgreich war.

Zur Programmierung des I²C-Busses **in Python** muss das Raspbian-Paket `python-smbus` installiert sein (oder werden):

```
pi@raspberrypi ~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Bild 2. i2cdetect ermittelt den LM75 unter der Adresse 48.

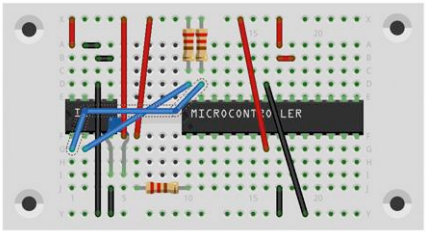


Bild 3. Der ATmega88 zusammen mit dem Sensor auf einem Steckbrett.

```
sudo apt-get install python-smbus
```

Danach kann man die Temperatur aus dem LM75 lesen:

```
#! /usr/bin/python
import smbus
import time

bus = smbus.SMBus(1)
address = 0x48
w = bus.read_word_data(address, 0)
print format(w, '04x')
```

Die „0" beim Aufruf der `read_word_data`-Methode gibt die Registernummer an, also hier das Temperaturregister (beim auf I²C aufsetzenden SMBus-Protokoll wird immer zuerst eine Registernummer übertragen). Das bereitet zum Beispiel beim Porterweiterungsbaustein PCF8574 Probleme, der ja kein Adressregister besitzt!

Beim ausgelesenen Wert hat man dasselbe Problem wie bei `i2cget`: Die beiden Bytes im Wort `w` sind vertauscht.

Listing 1. Main-Schleife zum Auslesen des LM75.

```
# include <i2cmaster.h>

# define LM75 (0x48 << 1)
                              // see datasheet
int
main(void)
{
unsigned char val[2];

i2c_init();
                              // initialisation I2C

i2c_start(LM75 | I2C_WRITE);
                              // addressing to write
i2c_write(0x00);
                              // temperature register
i2c_rep_start(LM75 | I2C_READ);
                              // addressing to read
val[0] = i2c_read();
                              // degrees Celsius
val[1] = i2c_read();
                              // tenth part bit
i2c_stop();
                              // ready
for(;;);
}
```

ATmega

Die ATmega-Mikrocontroller von Atmel verfügen über einen eingebauten I²C-Controller, der den Standard-Modus (100 kHz) und den Fast-Modus (400 kHz) beherrscht und als Master, als Slave und auch gemischt betrieben werden kann. Der ATmega324PB und der ATmega328PB besitzen sogar zwei I²C-Busse. In den folgenden Beispielen gehe ich von einem ATmega88 aus, den man wie in **Bild 3** zu sehen zusammen mit dem Sensor auf dem Steckboard unterbringt.

Hier benutzt man am besten die `i2cmaster`-Bibliothek von Peter Fleury [1], wobei zu beachten ist, dass diese Library die internen Pull-Up-Widerstände nicht aktiviert! Man muss also selber für die Pull-Up-Widerstände sorgen. In Bild 3 sind diese oben (zur 5-V-Schiene) eingezeichnet. Die Bibliothek stellt den I²C-Controller im ATmega auf den Standard-Modus (100 kHz) ein. Ein Zugriff auf den LM75-Sensor sieht dann so aus wie in **Listing 1**.

Mit der `lcdlibrary` vom selben Autor kann man dann mit nur einem zusätzlichen LC-Display ein digitales Thermometer aufbauen. Man kann auch eine I²C-Interface-Platine nutzen (auf der häufig ein PCF8574 verbaut ist), oder man verwendet gleich ein LCD mit I²C-Anschluss.

Der integrierte I²C-Controller der ATmega-Serie kann nicht nur im Master-Modus betrieben werden, um damit einen Slave anzusteuern, sondern auch im Slave-Modus, um dann von einem Master angesteuert zu werden. Sogar ein gemischter Master- und Slave-Betrieb ist möglich: Der ATmega kann als Master mit einem LM75 kommunizieren und dann von einem Raspberry Pi als Slave ausgelesen werden. Sinnvoll kann der Einsatz eines ATmega als Slave sein, wenn man Daten in Echtzeit über die Portpins des ATmega einliest und diese vorverarbeitet, um sie auf Nachfrage an einen Raspberry Pi weiterzuleiten.

Diese Art der Ansteuerung ist aber komplexer und die Interaktion mit dem Rest des Programms enger. Es gibt schon Ansätze einer Bibliothek, die den Slave-Betrieb implementiert [2].

Der nachfolgend beschriebene Ablauf folgt den in den Atmel-Datenblättern beschriebenen Zuständen. Entsprechende Tabellen finden sich in jedem ATmega-Datenblatt, so auch im Kapitel 22.7 des Datenblatts zum ATmega48/88/168.

Der Ablauf des Codes verläuft sinnvollerweise interruptgesteuert. Die I²C-Contol-Unit des ATmega löst einen Interrupt unter folgenden Bedingungen aus:

- nach dem Senden einer *START condition* beziehungsweise der *Repeated Start condition*,
- nach dem Senden der Adresse und des Schreib/Lese-Bits,
- nach dem Senden eines Datenbytes,
- wenn der ATmega die Arbitrierung verloren hat (beim Senden der *START condition* oder des Adress-Bytes oder der Schreib-/Lese-Bits ist es zu einer Kollision gekommen),
- wenn der ATmega eine *START condition* erkannt hat und als Slave adressiert wurde,
- wenn der ATmega ein Datenbyte empfangen hat,
- wenn der ATmega eine *STOP condition* oder eine *Repeated*

Start condition erkannt hat und (wieder) als Slave adressiert wurde oder
- wenn eine ungültige Bus-Transaktion erkannt wurde.

Die ersten vier Bedingungen betreffen den Master-Modus, somit interessieren uns hier lediglich die letzten drei Bedingungen.
Die I²C-Control-Unit muss mit der gewünschten Slave-Adresse initialisiert werden. Danach kann man sie einschalten und der normalen Arbeit nachgehen:

```
TWAR = (I2C_Slave_Addr << 1);
TWCR = _BM(TWEA) | _BM(TWEN) |
_BM(TWIE);
```

Die Bitrate muss im Slave Modus nicht eingestellt werden, da die Geschwindigkeit ja vom Master vorgegeben wird.
Im Falle eines Interrupts gilt es, den Grund dafür herauszufinden. Dazu liest man das Status-Register `TWSR` und wertet die höherwertigen fünf Bits aus.
In **Listing 2** ist ein Teil der Interrupt-Service-Routine (ISR) für die Adressierung als Empfänger, als Sender sowie die *STOP condition* zu sehen.
Die Beschreibungen und Tabellen in den ATmega-Datenblättern sind sehr ausführlich, sowohl was die Erklärung des Status-Codes als auch die erforderlichen Aktivitäten des Programms und die dadurch ausgelösten Aktionen der Hardware betrifft.

Arduino

Da viele Arduinos auf den Atmel-Mikrocontrollern der ATmega-Serie basieren, gelten eigentlich dieselben Ausführungen wie oben beschrieben. Allerdings gibt es für den Arduino eine oft genutzte und sehr praktische Bibliothek namens `Wire.h`.
Beim Arduino ist der Anschluss ähnlich unkompliziert wie beim Raspberry Pi (**Bild 4**). Je nach Arduino-Typ (oder Clone) ist auch hier darauf zu achten, dass ein mit 3,3 V betriebener Arduino nicht mit 5 V an den Ports konfrontiert werden sollte. Also erst messen, dann anschließen und gegebenenfalls die Pull-Up-Widerstände auf dem Slave entfernen!

Die `Wire`-Bibliothek ist in der Arduino-IDE schon enthalten; ähnlich wie bei der `i2cmaster`-Bibliothek von Peter Fleury muss die Bibliothek beziehungsweise die I²C-Schnittstelle aber erst initialisiert werden:

Listing 2. ISR zur Adressierung und Erkennung der *STOP condition*. Der vollständige und ausführlich kommentierte Code ist auf der Projektseite [5] zu finden.

```
ISR(TWI_vect)
{
    /*
     * These variables need to be preserved across interrupts
     */
    static unsigned char i2c_idx,       /* Index into twi_msg[] */
                         i2c_tosend;    /* Number of bytes to send */

    switch (TWSR & 0xf8)
    {
    ...
    /*
     * RECEIVE Code
     * See Table 19-4.  Status Codes for Slave Receiver Mode
     * [Page 229]
     */
    case 0x60:
        /*
         * Own SLA+W has been received; ACK has been returned
         * TWDR: No TWDR action
         * STA=X STO=0 TWINT=1 TWEA=1
         * Data byte will be received and ACK will be returned
         */
        TWCR = (TWCR & ~_BM(TWSTO)) | (_BM(TWINT) | _BM(TWEA));
        i2c_idx = 0;
        break;
    …
    case 0xA0:
        /*
         * A STOP condition or repeated START condition has been
         * received while still addressed as slave
         * TWDR: No action
         * TWA=0 STO=0 TWINT=1 TWEA=1
         * Switched to the not addressed Slave mode;
         * own SLA will be recognized;
         * GCA will be recognized if TWGCE = "1"
         */
        TWCR = (TWCR & ~(_BM(TWSTA) | _BM(TWSTO))) | (_BM(TWINT) | _BM(TWEA));
        break;
    …
    case 0xA8:
        /* Own SLA+R has been received; ACK has been returned
         * TWDR: Load data byte
         */
        /*
         * The address (register number) has been received,
         * Start sending payload
         */
      TWDR = 0x42;
      break;
    }
}
```

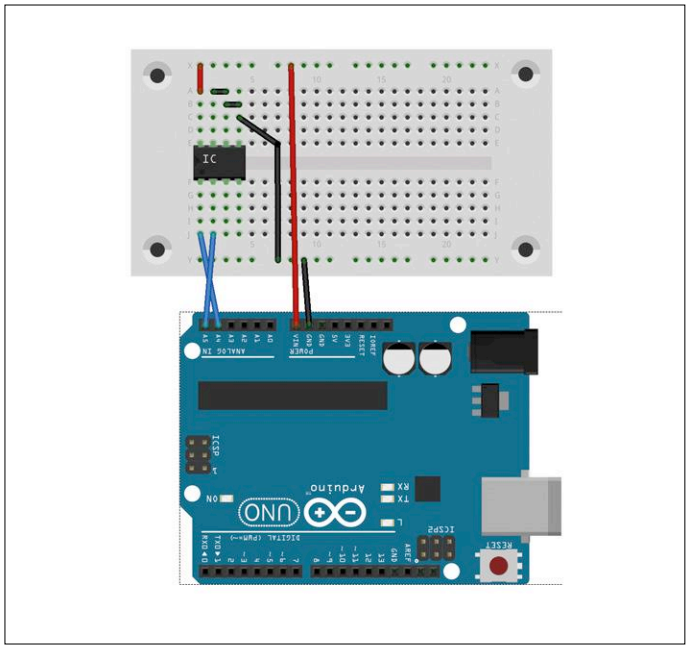


Bild 4. Der Aufbau mit dem Arduino ist fast identisch mit der RPi-Schaltung.

```
#include <Wire.h>
void setup() {
 Wire.begin();
}
```

Mit dem Aufruf von `Wire.begin()` werden die internen Pull-Up-Widerstände aktiviert. Sollte dies zu Problemen führen, weil die Widerstände doch recht groß dimensioniert sind, kann man extern zwei Widerstände von 10...20 kΩ parallel schalten. Oder man schaltet die internen Widerstände direkt wieder aus und verwendet nur externe (dann mit etwa 4,7 kΩ). Dies kann man mit den folgenden zwei Zeilen nach dem Aufruf von `Wire.begin()` erledigen:

```
digitalWrite(SDA, 0);
digitalWrite(SCL, 0);
```

Anmerkung: Dies ist nicht offiziell und in Zukunft könnte auch alles anders sein. Am besten misst man nach der Initialisierung mit einem Multimeter nach.
Achtung: Auch wenn man die internen Pull-up-Widerstände direkt abschaltet, so sind sie doch kurzzeitig aktiv. Dies kann zu gravie-

I²C-Bus am PC

Sogar jeder PC hat sein I²C-Interface, meist sogar mehrere davon. I²C-Slaves stecken in Displays (DDC [3]) und auf Speicherriegeln (SPD [4]). Auch können Temperatursensoren über I²C angeschlossen sein. Leider geben nur wenige bis gar keine Hersteller Informationen heraus, welche I²C-Bausteine eingebaut sind und ob und wie man von außen an die I²C-Busse gelangen kann (wenn diese denn überhaupt von außen erreichbar sind und sich nicht nur auf irgendwelchen Modulen befinden).
Wenn man Linux einsetzt, kann man einfach mal probieren, das `i2c-dev`-Modul zu laden:

```
$ sudo modprobe i2c-dev
```

und danach schauen, ob es Geräteknoten in `/dev` gibt:

```
$ ls /dev/i2c*
/dev/i2c-0 /dev/i2c-1 /dev/i2c-2 /dev/
i2c-3 /dev/i2c-4 /dev/i2c-5
```

Letzteres geht auch ohne Root-Berechtigung. Unter Debian und seinen Derivaten (Raspbian und Ubuntu) gibt es die `i2c-tools`:

```
apt-get install i2c-tools
```

Manchmal ist nur einer der Busse mit Slaves bestückt:

```
root@bounty:~# i2cdetect -y 5
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- 37 -- -- 3a -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- 58 -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Es ist jedoch nicht immer klar, um welche Bausteine es sich handelt. Hier liegt auf Adresse `0x50` das EDID-PROM des Displays:

```
root@bounty:~# i2cdump -y 5 0x50
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 ff ff ff ff ff ff 00 1a b3 d4 07 ec 22 02 00    ........?????"?.
10: 0a 16 01 03 80 34 20 78 2a ef 95 a3 54 4c 9b 26    ?????4 x*???TL?&
20: 0f 50 54 a5 4b 00 81 80 81 00 81 0f 95 00 95 0f    ?PT?K.???.???.??
30: a9 40 b3 00 01 01 28 3c 80 a0 70 b0 23 40 30 20    ?@?.??(<??p?#@0
40: 36 00 06 44 21 00 00 1a 00 00 00 fd 00 38 4c 1e    6.?D!..?...?.8L?
50: 52 10 00 0a 20 20 20 20 20 20 00 00 00 fc 00 42    R?.?      ...?.B
60: 32 34 57 2d 35 20 45 43 4f 0a 20 20 00 00 00 ff    24W-5 ECO?  ....
70: 00 59 56 32 45 31 34 30 30 31 32 0a 20 20 00 8e    .YV2E140012?  .?
```

Wer noch einen VGA-Anschluss hat, kann versuchen, an Pin 12 (SDA) und Pin 15 (SCL) einen I²C-Slave anzuschließen. Pin 6 (SCL) und Pin 7 (SDA) einer DVI-Buchse und Pin 15 (SCL) und Pin 16 (SDA) einer HDMI-Buchse sind auch gute Kandidaten für Experimente. Eventuell sind diese Busse aber auch vollständig unter der Kontrolle der Grafikkarte und der Firmware darauf.

renden Problemen führen, wenn man einen 3,3-V-Slave mit zwei externen Pull-Up-Widerständen nach 3,3 V an den 5-V-Arduino anschließt. In einem solchen Fall muss unbedingt ein Pegelwandler verwendet werden! Nach der Initialisierung kann man in der `loop()`-Funktion Daten an den I²C-Slave schicken, zum Beispiel die Register-Adresse des Temperaturregisters des LM75:

```
void loop() {
 Wire.beginTransmission(0x48);
 Wire.write(byte(0x00));
 Wire.endTransmission();
```

oder von ihm empfangen:

```
Wire.requestFrom(0x48, 2);
 c1 = Wire.read();
 c2 = Wire.read();
}
```

Wie man sieht, muss man bei der `Wire`-Bibliothek angeben, wie viele Bytes man vom Slave empfangen will (hier: `2`). Mit diesem Aufruf liest man die beiden Bytes schon ein. Danach können sie einzeln abgerufen werden.
Die Arduino `Wire`-Bibliothek kann auch für den Slave-Modus verwendet werden. Auch hier muss erst initialisiert werden. Dazu wird `Wire.begin()` mit einem Parameter aufgerufen, der die gewählte Slave-Adresse spezifiziert. Dadurch wird der Slave-Modus ausgewählt. Zusätzlich muss ein Event-Handler angegeben werden, der aufgerufen wird, wenn der Arduino als Slave angesprochen wird, in diesem Beispiel für den Fall, dass der Arduino Daten empfangen soll:

```
#include <Wire.h>
void setup() {
 Wire.begin(0x48);
 Wire.onReceive(receive);
}
```

Auch hier werden, für einen Slave eher unüblich, in der `Wire.begin()`-Methode die Pull-Up-Widerstände eingeschaltet. Man sollte sie wie oben gezeigt direkt wieder abschalten.
Warnung: Damit wäre ein 5-V-Arduino zwar auch als Slave an einem 3,3-V-Master einsetzbar, jedoch kann ein kurzzeitiger 5-V-Impuls auf den Leitungen einen 3,3-V-Master zerstören. Will man also einen Arduino als Slave beispielsweise an einem Raspberry Pi betreiben, muss man unbedingt sicherstellen, dass der Arduino erst nach der Initialisierung elektrisch mit dem Raspberry Pi verbunden wird.
Sobald nun Daten eintreffen, wird der oben genannte `receive`-Event-Handler aufgerufen. Auch hier sind die Daten bereits gelesen; als Parameter wird die Anzahl der empfangenen Bytes übergeben:

```
void receive(int n) {
 while (n--) {
  uint8_t c = Wire.read();
  // Verarbeite c
 }
}
```

Für die Übertragung von Daten zum Master ruft man statt `onReceive()` die Methode `onRequest()` auf und übergibt ihr einen Event-Handler, der die Daten bereitstellt. Da hier nicht bekannt ist, wie viele Daten erwartet werden beziehungsweise wie viele Daten geschickt werden, hat der Event-Handler auch keinen Parameter. Die Daten müssen jedoch in **einem** `Wire.write()`-Aufruf übergeben werden:

```
#include <Wire.h>
void setup() {
 Wire.begin(0x48);
 Wire.onRequest(transmit);
}
void loop() {
 while (1) delay(1000);
}
void transmit() {
 uint8_t msg[N];
 // Erzeuge msg[] Inhalt
 Wire.write(msg, N);
}
```

Beide Methoden lassen sich auch zusammen in einem Sketch verwenden:

```
#include <Wire.h>
void setup() {
 Wire.begin(0x48);
 Wire.onReceive(receive);
 Wire.onRequest(transmit);
}
```

Damit ließen sich dann ein genauer LM75 nachbilden, 1-Wire-Temperatursensoren (DS18B20) als LM75-Ersatz einsetzen oder eine RTC emulieren, die die Uhrzeit von DCF77 oder GPS erhält.

Im dritten Teil

Der folgende Teil der Serie stellt einige beliebte I²C-Komponenten vor, neben dem Temperatursensor LM75 auch den Port-Expander PCF8574 und die Echtzeituhr RV8523. Die kleine Artikelreihe wird vom Thema Fehlersuche und -analyse mit geeigneten und für das Heimlabor erschwinglichen Hilfsmitteln abgeschlossen. ⏮

(160418)

Weblinks

[1] http://homepage.hispeed.ch/peterfleury/avr-software.html
[2] www.mikrocontroller.net/topic/182638
[3] https://en.wikipedia.org/wiki/Display_Data_Channel
[4] https://en.wikipedia.org/wiki/Serial_presence_detect
[5] www.elektormagazine.de/160148

Wirklich gedruckte Schaltungen

Platinendrucker/Reflow-Ofen Voltera V-One

Von **Dipl.-Inf. Karl-Ludwig Butte** (D)

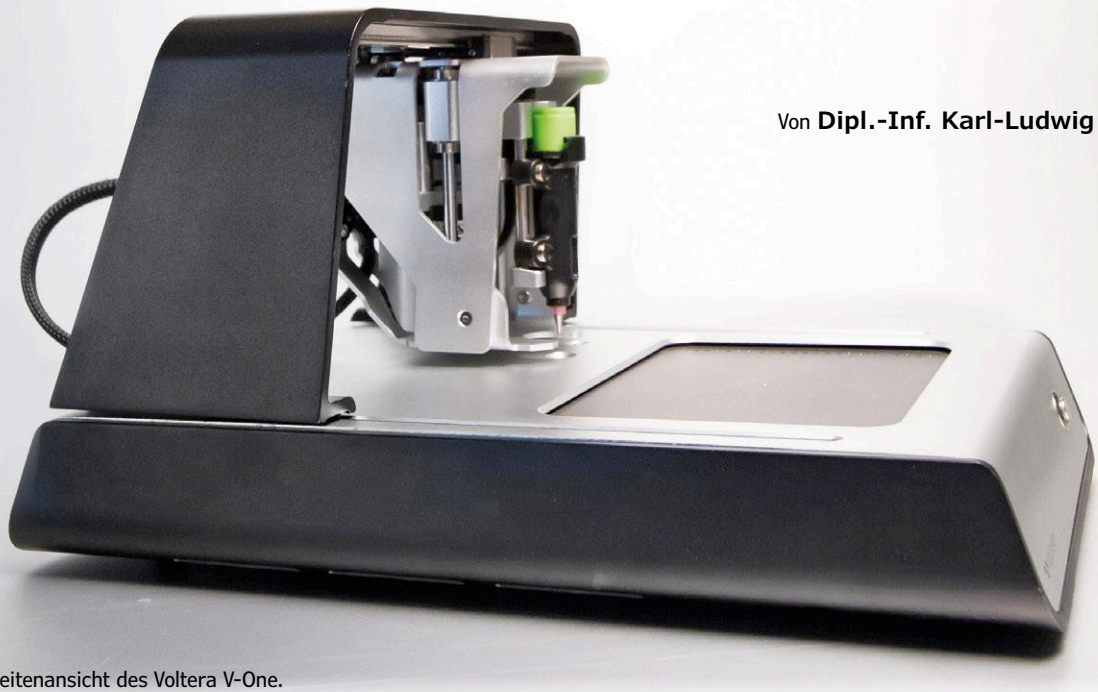


Bild 1. Seitenansicht des Voltera V-One.

Was war es doch früher für eine Chemikalien-Panscherei, wenn man eine Platine selbst ätzen wollte: Zuerst musste das Platinen-Layout fototechnisch auf eine Klarsichtfolie kopiert werden - also belichten, entwickeln und fixieren. Und dann ging es nur so weiter mit viel Chemie, Flecken und Gestank. Doch jetzt könnte alles anders werden: Der Voltera V-One druckt Leiterbahnen, trägt Lötpaste auf und kann die Bauteile sogar im Reflow-Verfahren löten.

Das Hantieren mit diversen Chemikalien hat viele Entwickler dazu bewogen, Platinen für Prototypen nicht mehr selbst herzustellen. Schon bei der Belichtungsfolie ging es los: Per Trial & Error mussten Belichtungs- und Entwicklungszeiten ermittelt werden. Das Ziel war, den besten Kompromiss zwischen dem Deckungsgrad der schwarzen Leiterbahnen und der Durchlässigkeit der umgebenden Flächen zu ermitteln. Je lichtundurchlässiger die Leiterbahnen, desto grauer waren die umgebenden Flächen, wodurch dann Kupfer da nicht vollständig weggeätzt wurde, wo keines sein sollte. Waren umgekehrt die umgebenden Flächen schön durchsichtig, dann fiel der Deckungsgrad der Leiterbahnen nicht hoch genug aus und gerade feine Leiterbahnen wurden teilweise ebenfalls weggeätzt. War der Platinenfilm in akzeptabler Qualität fertiggestellt, konnte es endlich losgehen. Nun wurde die Platine belichtet, entwickelt und geätzt. Ging alles gut, wurde man nach dem Entfernen der ätz-resistenten Fotoschicht mit glänzenden Kupferbahnen belohnt. Statt von „Printed Circuit Boards“ bzw. „gedruckten „Schaltungen“ hätte man eigentlich von „Etched Circuit Boards“, also ECBs sprechen müssen.

Nun könnte das kanadische Startup Voltera mit seinem Platinen-Drucker V-One die Platinenherstellung revolutionieren und wieder mehr Elektroniker dazu bewegen, Platinen selbst herzustellen. Eine Technik wie geschaffen für das 21. Jahrhundert, denn mit dem V-One kann man nicht nur die Platine drucken, sondern auch die Lötpaste auftragen und manuell platzierte Bauteile im Reflow-Verfahren löten.

Die Hardware

Beim V-One handelt es sich um einen Flachbettdrucker mit Drei-Achsen-Steuerung, der bequem auf dem Schreibtisch Platz findet (**Bild 1**). Der mechanische Aufbau ist in **Bild 2** zu sehen: Eine Basis, auf der eine bewegliche Brücke montiert ist (Y-Achse). An dieser Brücke befindet sich die Aufnahme für drei unterschiedliche Werkzeuge. Sie überstreicht die X-Achse. Zusätzlich kann das Werkzeug noch in der Höhe verstellt werden

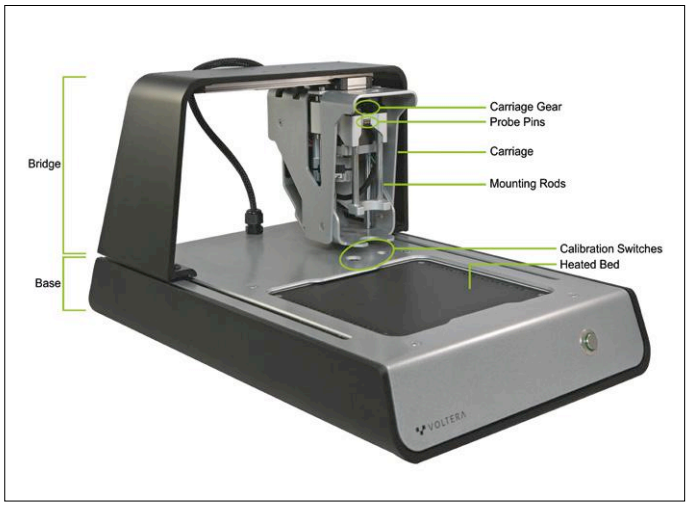


Bild 2. Mechanischer Aufbau des V-One.

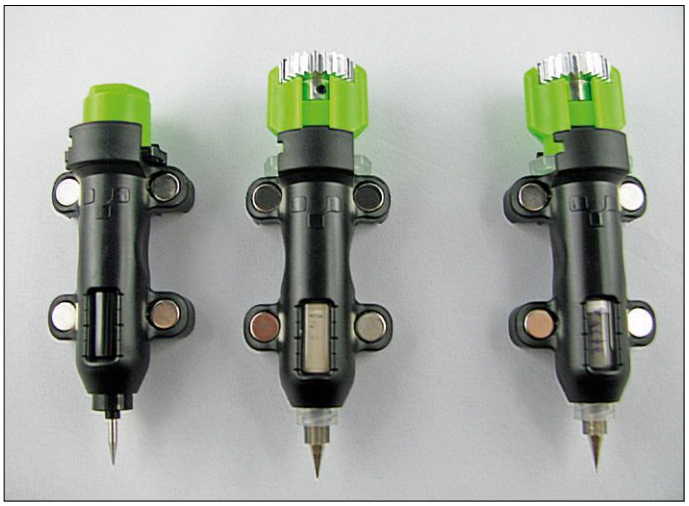


Bild 3. Werkzeuge des Voltera V-One.

(Z-Achse).
Bei den drei Werkzeugen von **Bild 3** handelt es sich um einen Tastkopf, einen Druckkopf für die elektrisch leitfähige Tinte und einen Druckkopf für Lötpaste. Die Werkzeuge lassen sich leicht wechseln, da sie nur mit vier starken Magneten befestigt werden.
Das zu bedruckende Platinen-Material (FR4) wird mit zwei Klemmschienen auf der Arbeitsfläche fixiert. Unter dieser Arbeitsfläche ist ein Heizsystem montiert, mit dem nicht nur die gedruckten Leiterbahnen fixiert, sondern in einem weiteren Arbeitsschritt auch SMD-Bauteile reflow-gelötet werden können. Mit dem V-One druckt man also nicht nur die eigentliche Platine, sondern macht auch noch einen extra Reflow-Ofen überflüssig! Da der V-One auch noch die Lötpaste präzise auf die Pads aufträgt, braucht es keine Stencils mehr. Mit dem V-One kann jeder Elektroniker SMD-Bauteile einfach und dennoch professionell verarbeiten!
Die Schlitze, in denen die Brücke entlang der Y-Achse geführt wird, werden von unten durch zwei Reihen RGB-LEDs beleuchtet (**Bild 4**). Das ist nicht nur ein Show-Effekt, sondern signalisiert die Temperatur der Arbeitsfläche: Rot bedeutet „Finger weg!“. Die Arbeitsplatte kann jetzt bis zu 250 °C heiß werden. Leuchtet es blau, ist die thermische Phase eines Brenn- bzw. Reflow-Prozesses abgeschlossen, und man kann die Platine entfernen. Der Drucker verfügt über eine USB-Schnittstelle, für die zumindest unter Microsoft Windows ein spezieller, von der Voltera-Internetseite downloadbarer Treiber installiert werden muss.

Die Software

Die zugehörige Software ist eine Gesamtlösung in Form einer *Kalibrierungs-Steuerungs-Druck-Videobedienungsanleitungs-Software*. Besonders der letzte Punkt verdient Lob: Jeder Arbeitsschritt wird an der rechten Bildschirmseite mit ein oder mehreren kurzen Videos erläutert und jeder Handgriff praktisch vorgeführt (**Bild 5**). Diese Videos laufen endlos und helfen beim Nachvollziehen der gezeigten Handgriffe am eigenen Gerät. Einfacher und deutlicher geht es kaum!
Die Software ist für Microsoft Windows 7 (64 Bit) oder höher und für OSX ab Version 10.11 verfügbar. Eine Linux-Version lässt leider noch auf sich warten. Die Software gibt es stets aktuell per Download – auf mittlerweile antike optische Medien

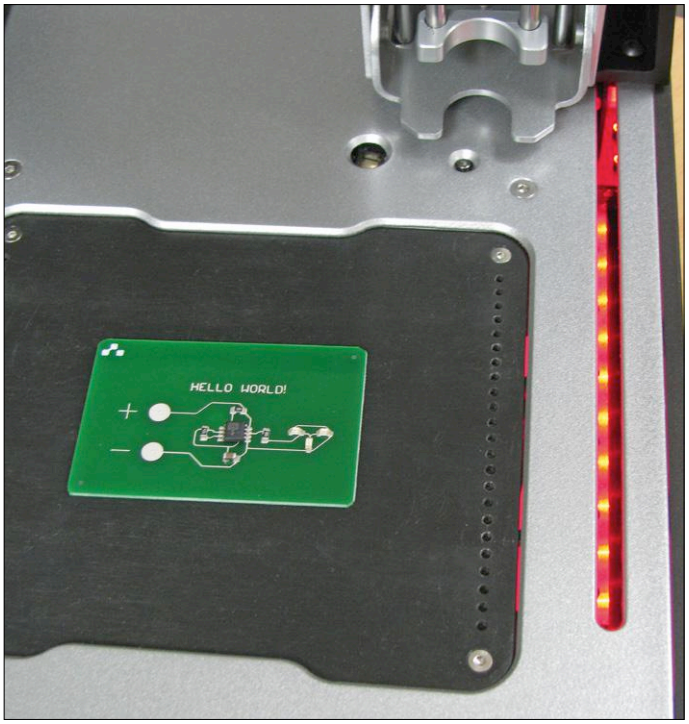


Bild 4. Farbige Betriebsmodi: Leuchten die Schlitze der Y-Achse rot, ist die Arbeitsplatte sehr heiß.

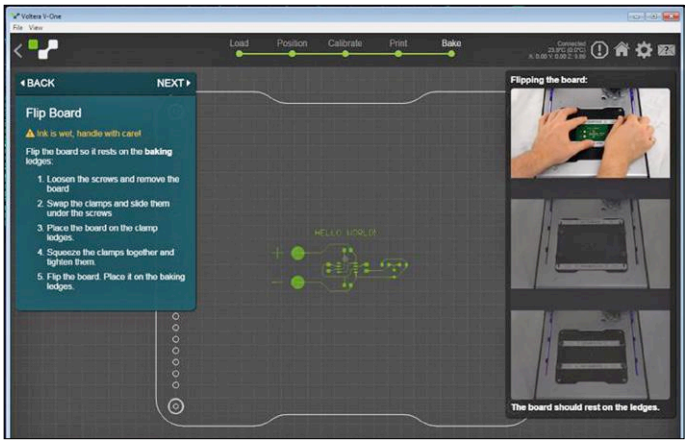


Bild 5. Die Software zum V-One enthält erklärende Video-Tutorials.

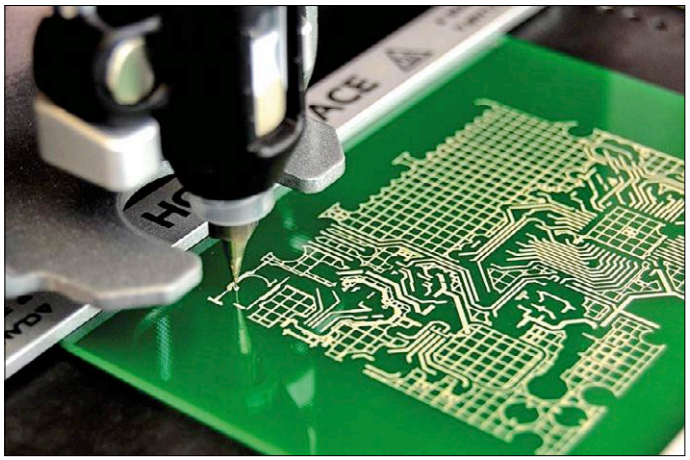


Bild 6. V-One bei der Arbeit: gedruckte Leiterbahnen.

Bild 7. Temperaturverlauf (Soll- und Istwerte) beim Backen der Leiterbahnen.

wurde verzichtet. Bei jedem Start der Software wird zudem eine Überprüfung auf Updates durchgeführt. Bei Vorliegen werden die Software und/oder die Drucker-Firmware automatisch installiert.

Arbeiten mit dem V-One

Der Voltera V-One verarbeitet ausschließlich Gerber-Dateien, die ja mit den meisten heutigen Platinen-Layout-Programmen erzeugt werden können. Voltera hat die Gerber-Dialekte von Eagle, Altium, Upverter und KiCad getestet. Auch bei Elektor-Projekten stehen ja vielfach Gerber-Dateien zum Download zur Verfügung. Um den Drucker sofort ausprobieren zu können, legt Voltera netterweise alle benötigten Bauteile für eine kleine „Hello World-Schaltung“ bei. Dabei handelt es sich um eine Blink-Applikation mit dem allseits bekannten Timer NE555 als SMD. Das Platinen-Layout ist sogar schon in der Software integriert und über einen besonderen Link im Datei-Öffnen-Dialog erreichbar.

Drucken

Als Platinenmaterial eignen sich die üblichen glasfaserverstärkten Epoxyd-Platten des Typs FR4. Meinem V-One lag nicht nur je ein Zehnerpack Platinen mit den Abmessungen 2 x 3“ und 3 x 4“ bei, sondern auch je ein Shield für einen Arduino Uno und Mega. Die maximale Druckfläche beträgt 138 x 102 mm (5,5 x 4“). Es hat sich bei mir bewährt, vor dem Druck die Platinen mit Spiritus zu reinigen, da Spuren von Hautfett zu unterbrochenen Leiterbahnen führen können.

Als erstes wird die noch rohe Platine in der Mitte der Arbeitsfläche mit den beiden Metallbügeln fixiert. Im nächsten Schritt fährt der Tastkopf das Rechteck über der Platine ab, innerhalb dessen die Schaltung gedruckt wird. Liegt das Rechteck nicht mittig oder außerhalb der Platine, kann man am Bildschirm die Lage solange korrigieren, bis alles passt. Auf diese Weise lassen sich auch mehrere kleinere Layouts nacheinander auf dieselbe Platine drucken, um sie dann später in Einzelplatinen zu trennen.

Ist die Lage der Platine korrekt, wird mit dem Tastkopf eine Höhen-Analyse durchgeführt. Die Druckspitzen für die elektrisch leitfähige Tinte und die Lötpaste sind hauchdünn und sehr empfindlich. Sie dürfen auf keinen Fall mit der Platinenoberfläche kollidieren, aber auch nicht zu hoch über ihr schweben, da sonst die Qualität des Drucks darunter leiden würde.

Nach abgeschlossener Höhenanalyse wird der Tastkopf gegen den Druckkopf mit elektrisch leitfähiger Tinte ausgetauscht. Damit wird zunächst ein Kalibrierungsmuster gedruckt: Zwei Schlangenlinien und mehrere waagerechte Parallelen im Minimalabstand von 0,8 mm. Die Leiterbahnen haben eine Minimalstärke von 8 mil. (~0,2 mm). Wird dieses Muster nicht ordnungsgemäß gedruckt, kann man mit der Software den Tintenfluss anpassen und - nach Abwischen des alten Musters - die neue Einstellung mit erneutem Druck überprüfen. Laut Herstellerangabe müssen die elektrisch leitfähige Tinte und die Lötpaste im Kühlschrank gelagert und 30 Minuten vor dem Druck an die Zimmertemperatur angepasst werden. Bei mir hat sich allerdings eine wesentlich längere Aufwärmzeit von ca. 2 h bewährt, da damit das Druckergebnis viel gleichmäßiger und vor allem lückenfrei ausfiel. Ist das Kalibrierungsmuster okay, wird es mit Küchenpapier weggewischt und die Platine nochmals mit Spiritus gereinigt.

Es ist immer wieder faszinierend, dem Drucker bei der Arbeit zuzuschauen (**Bild 6**). Dabei wird der Druckkopf nicht wie bei Tintenstrahl-Druckern zeilenweise über die Platinenoberfläche geführt, sondern es werden wie bei einem Plotter ganze Leiterbahnen am Stück gedruckt und anschließend mit Löt-Pads komplettiert.

Sollten einige Stellen nicht zufriedenstellend gedruckt worden sein (das kann an vielfältigen Gründen liegen), kann man den Druck von am Bildschirm ausgewählten Stellen wiederholen und dabei gegebenenfalls die Tintenzufuhr korrigieren.

Backen

Ist der Druck vollständig abgeschlossen, kann der Back- oder Fixier-Prozess beginnen, um die noch feuchten Leiterbahnen auszuhärten. Solange ist also noch Vorsicht geboten, damit man nicht aus Versehen die Tinte verwischt. Zum Backen werden die Haltebügel vertauscht und die Platine mit der bedruckten Seite nach unten in die Vertiefung der Haltebügel gelegt, sodass die Oberfläche der Platine mit etwas Abstand über der Arbeitsfläche gehalten wird.

Der Backprozess wird von der Software angestoßen und dauert ca. 40...50 Minuten. Währenddessen werden fortlaufend Soll- und Ist-Temperatur des Heizsystems grafisch am Bildschirm angezeigt (**Bild 7**). Jetzt „drohen“ die Führungsschlitze der Brücke in tiefem Rot: Das Berühren des Druckers ist jetzt nicht

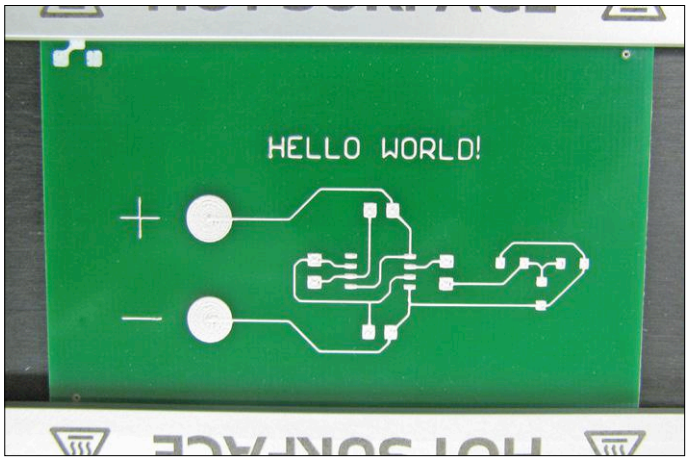


Bild 8. Nach dem Backen der Leiterbahnen: Die fertig gedruckte und polierte Platine.

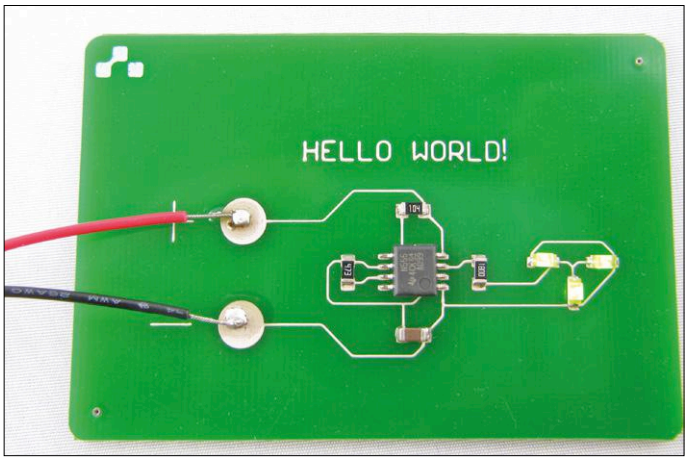


Bild 9. Die fertige, vom V-One gelötete Platine ist bereit zum Test.

ratsam, denn 250 °C können böse Verbrennungen verursachen. Meinen V-One habe ich daher vorsichtshalber auf zwei Lagen Keramik-Kacheln gestellt. Leuchten die LEDs dann wieder blau, ist das Heizsystem abgekühlt und der Back-Prozess beendet. Man kann die Platine jetzt gefahrlos entnehmen.
Dem V-One liegt ein Spezial-Kunststoffschwamm bei, mit dem nun die Platine poliert werden muss, damit nachher sichere Lötverbindungen entstehen. **Bild 8** zeigt die fertig gedruckte und polierte Platine.

Lötpaste auftragen

Bevor mit dem Auftragen der Lötpaste begonnen werden kann, muss die Platine wieder auf der Arbeitsfläche fixiert werden; danach muss man dem Drucker die genaue Lage zweier Referenz-Punkte übermitteln. Dies geschieht problemlos mit entsprechenden Funktionen der Software und dem Tastkopf. Anschließend wird dieser gegen den Druckkopf für Lötpaste ausgetauscht und der Vorgang gestartet.

Bestücken und Reflow-Löten

Die Bestückung mit Bauteilen ist noch nicht automatisiert und wird von Hand vorgenommen. Eine gute Pinzette ist dafür unerlässlich. Trotzdem ist größte Vorsicht geboten, dass man nichts von der Lötpaste verwischt und gepolte Bauteile (Elkos, Dioden, ICs, etc.) richtig herum auflegt. Sind alle Bauteile an ihrem Platz, wird die Platine mittig auf die Arbeitsfläche aufgelegt und der Reflow-Prozess gestartet (Bild 4). Wenn die LEDs wieder blau leuchten, ist die Platine fertig und kann getestet werden (**Bild 9**). Die Anschlussdrähte des 9-V-Batterie-Clips links in Bild 9 wurden nachträglich angelötet, was uns zum letzten Thema dieses Artikels bringt.

Verarbeitung bedrahteter Bauteile

Auf die Frage, ob man mit dem V-One nur SMD-Bauteile verarbeiten kann, gibt es eine klare Antwort: Es lassen sich auch bedrahtete Bauteile verwenden. Dazu nutzt man im Platinen-Layout entsprechende Löt-Pads mit Loch in der Mitte. Statt dem automatisierten Auftragen von Lötpaste erfolgt dann nach dem Backen und Polieren der Platine das althergebrachte manuelle Bohren und Bestücken. Aber beim Löten muss man sich auf Änderungen einstellen, denn elektrisch leitfähige Tinte verhält sich anders als Kupfer! Man benötigt unbedingt eine temperaturgeregelte Lötstation, die auf 190 bis max. 210 °C eingestellt werden darf. Gut geeignet ist das von Voltera empfohlene Lötzinn (Sn62 oder noch besser SnBiAg1). Es ist so dünn, dass es keine Flussmittelseele hat. Das Flussmittel (z.B. ChipQuick SMD 291) muss vorher gesondert aufgetragen werden. Wie ich selbst feststellen musste, braucht es hierfür etwas Übung.

Platinen automatisch drucken und reflow-löten

Fazit

Mit dem Voltera V-One gehört das Platinenätzen endgültig der Vergangenheit an. Einmal ausprobiert wird man als Elektroniker so ein Gerät nicht mehr missen wollen. Gerade bei der Hardware-Entwicklung ist es ein großer Vorteil, nicht mehr Tage oder Wochen auf den Platinen-Prototypen eines Dienstleisters warten zu müssen. Der integrierte Reflow-Ofen ist das Tüpfelchen auf dem „i“ und lässt das Löten von SMDs zur Routine werden. Der Platinen-Drucker ist laut Preisliste von Voltera mit 2.179 $ zwar nicht ganz billig und die dazugehörigen Verbrauchsmaterialen genauso wenig, aber meiner Meinung nach jeden Cent wert. ◀

(160384)

Weblink

[1] www.voltera.io

Über den Autor:

Karl-Ludwig Butte studierte Informatik mit Fachrichtung Wirtschaft an der FH Fulda und bearbeitet zurzeit IT- und Prozessthemen in der Automobilindustrie. In seiner Freizeit befasst er sich neben Elektronik-Projekten auch mit Rechenmaschinen aller Art und der Geschichte des Computers.

nWatch, ein Wearable-Entwicklungssystem

Oder: Wie ich meine eigene SmartWatch baute

Von **Piotrek Wasilewski** (Polen)

Ursprünglich als einfacher Spickzettel für Klassenarbeiten gedacht, wurde diese Uhr ziemlich schnell ziemlich intelligent. Auf dem Weg zu einem wirklich sinnvollen Gerät lernte der Autor eine Menge. Also, ein cooles Gadget oder ein spaßiges Lernmittel oder beides? Bauen Sie die nWatch und finden Sie es heraus!

Eigenschaften

- ARM-Cortex-M4
- Bluetooth Low Energy (BLE)
- Gyro, Beschleunigungsmesser und Magnetometer
- MP3-Player
- 3D-gedrucktes Gehäuse
- Großartiges Lernprojekt

Die nWatch ist eine intelligente Uhr, die gleichzeitig als Mikrocontroller-Entwicklungsboard fungiert. Am Anfang war dieses Projekt aber nicht wirklich ein Entwicklungsboard, sondern vielmehr ein einfaches Mittel, um bei Klassenarbeiten zu schummeln, wenn es sich um – meiner Meinung nach – völlig irrelevanten Lernstoff handelte. Der erste Prototyp tat genau das, wofür er erschaffen wurde, war aber noch nicht besonders intelligent. Dann habe ich damit begonnen, daraus ein nützliches Gerät zu machen, eine intelligente Armbanduhr, mit der sich auch Mikrocontroller-Projekte entwickeln ließen.

Die Hardware der nWatch

Bild 1 zeigt das Blockschaltbild der nWatch. Alle diese Blöcke kann man im Schaltbild (**Bild 2**) wiederfinden, es sind nur wenige passive Bauteile hinzugekommen. Die Blöcke sollen nun in loser Folge besprochen werden.

Die CPU

Das Herz der nWatch ist IC1, ein Mikrocontroller STM32f407ZGT6 der leistungsstarken Foundation-Familie von STMicroelectronics. Er besitzt einen ARM-Cortex-M4-Kern, der mit 168 MHz getaktet wird, einen Digitalen Signalprozessor DSP und eine Gleitkommaeinheit (floating point unit, FPU). Es gibt 1 MB Flash-Speicher und eine ganze Galaxie von Peripherie. Der Controller wurde gewählt, weil er erstens ausreichend Allzweck-Ports (GPIO) und Interfaces besitzt, um alle anderen ICs anschließen zu können. Mit der speziellen Hardware-MMU (memory management unit) wird das LCD und der externe SRAM-Speicher IC10 gesteuert, viel einfacher, als dies mit den unabhängigen GPIO-Pins möglich gewesen wäre. Im Controller stecken auch ein schnelles RAM und ausreichend Flashspeicher für meine Zwecke. Der Controller hat aber auch einen Nachteil: Er ist ziemlich groß (relativ gesehen natürlich). Aber es gab keine andere Lösung, da ich keine CPU im Ball-grid-array-Gehäuse verwenden wollte.

Externes SRAM und MicroSD-Karte

Die nWatch verfügt über zwei externe Speicher, einen SRAM-Chip und eine SD-Karte. Das SRAM (IC10) war ursprünglich nicht geplant, aber es hat sich als notwendig erwiesen, um das Display bequem steuern zu können, da das interne RAM der CPU nicht groß genug für die STemWin-Grafikbibliothek ist (siehe unten). IC10 wird als Bildschirmpuffer verwendet. Alle Änderungen am Displayinhalt werden in diesem Speicher zwischengelagert, bevor sie zum Puffer des Displays übertragen werden. Dieser Speicher ist nicht so schnell wie das Controller-RAM. Daher werden alle zugeteilten Stacks (FreeRTOS-Speicher und so weiter) im internen Speicher gehalten, damit die CPU so schnell wie möglich arbeiten kann. Das externe SRAM ist gemeinsam mit dem LCD an einem Bus angeschlossen und wird vom flexiblen statischen Speichercontroller (FSMC) der CPU gesteuert. Der zweite externe Speicher ist die nichtflüchtige SD-Karte (K1). Eine SD-Karte lässt sich leicht entfernen und an einen Computer oder ein Smartphone anschließen, um ein neues Bild oder eine Textdatei hochzuladen. Der SD-Verbinder ist über einen schnellen SDIO-Bus angeschlossen, der ein viel schnelleres Lesen und Schreiben von Dateien als über SPI ermöglicht.

Das Display

Das Display aus einem alten Nokia-Handy des Typs 6300 ist schuld am Buchstaben „n“ in „nWatch“ (**Bild 3**). Ich habe es ausgewählt, weil es ganz einfach zu steuern ist, da es einen 8-Bit-Parallelbus benutzt. Darüber hinaus ist es billig (weniger als 5 €) und besitzt einen durchaus „bastelkompatiblen“ Verbinder. Es hat die richtige Größe, klein genug für

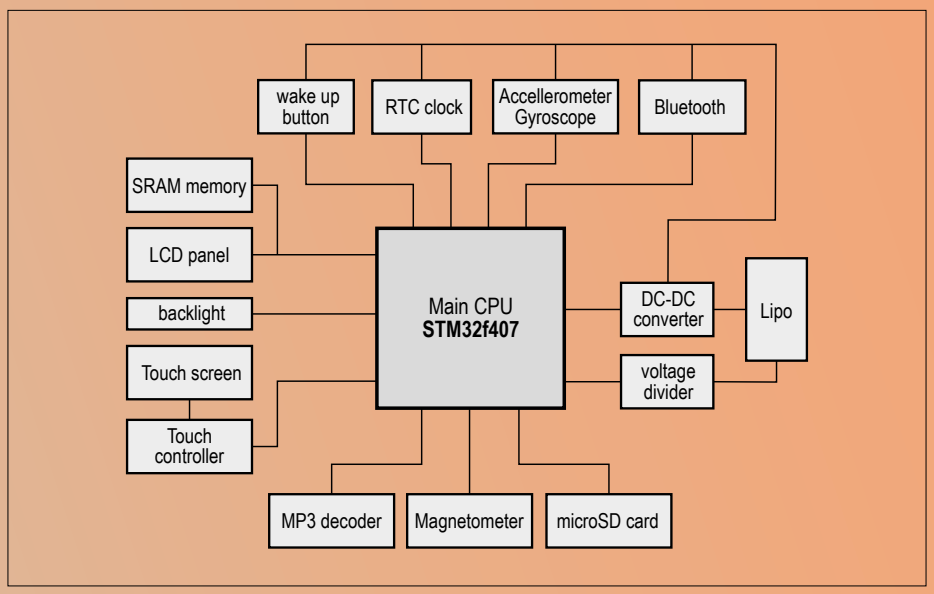


Bild 1. Die Innereien der nWatch waren nicht einfach in einer Armbanduhr unterzubringen.

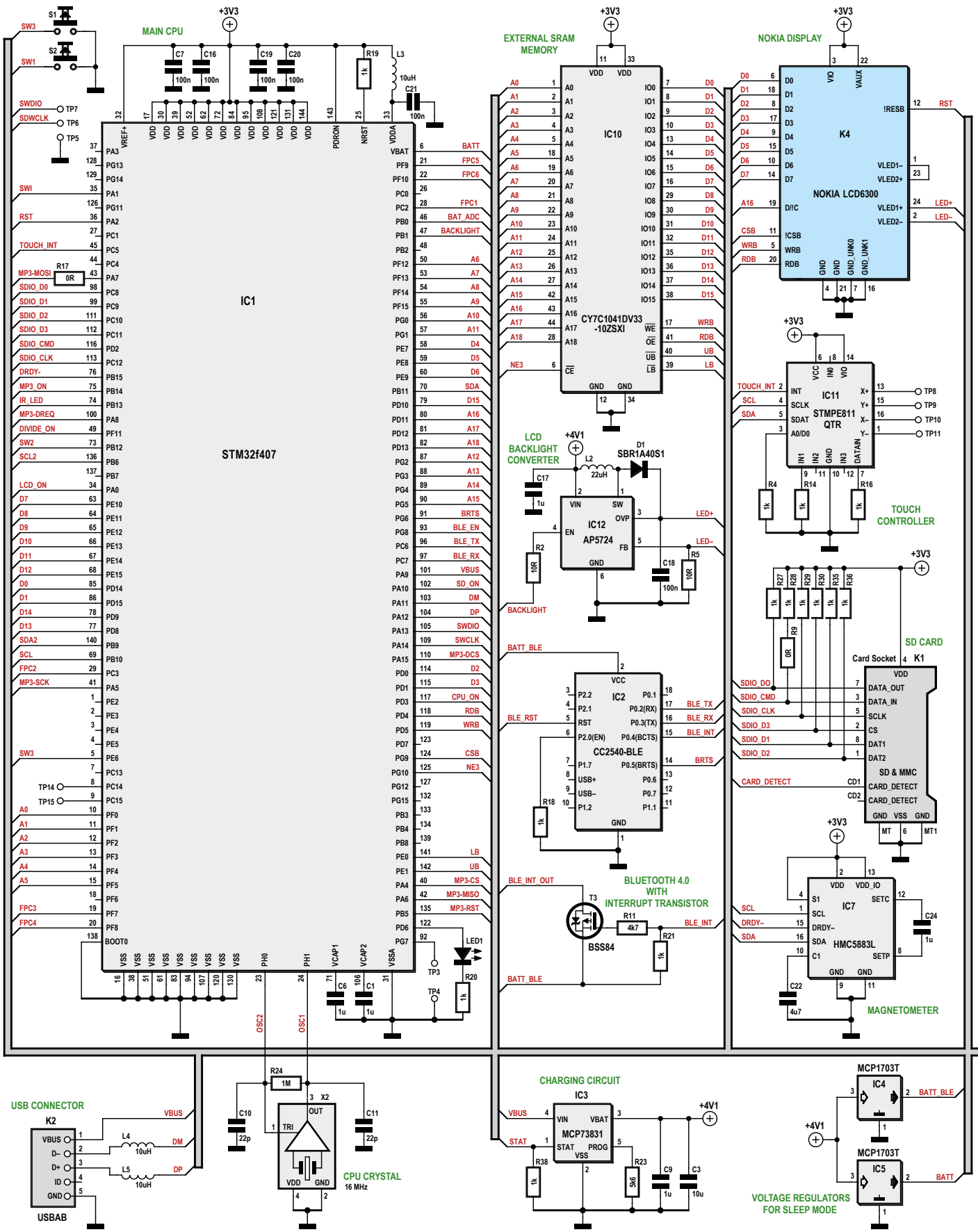
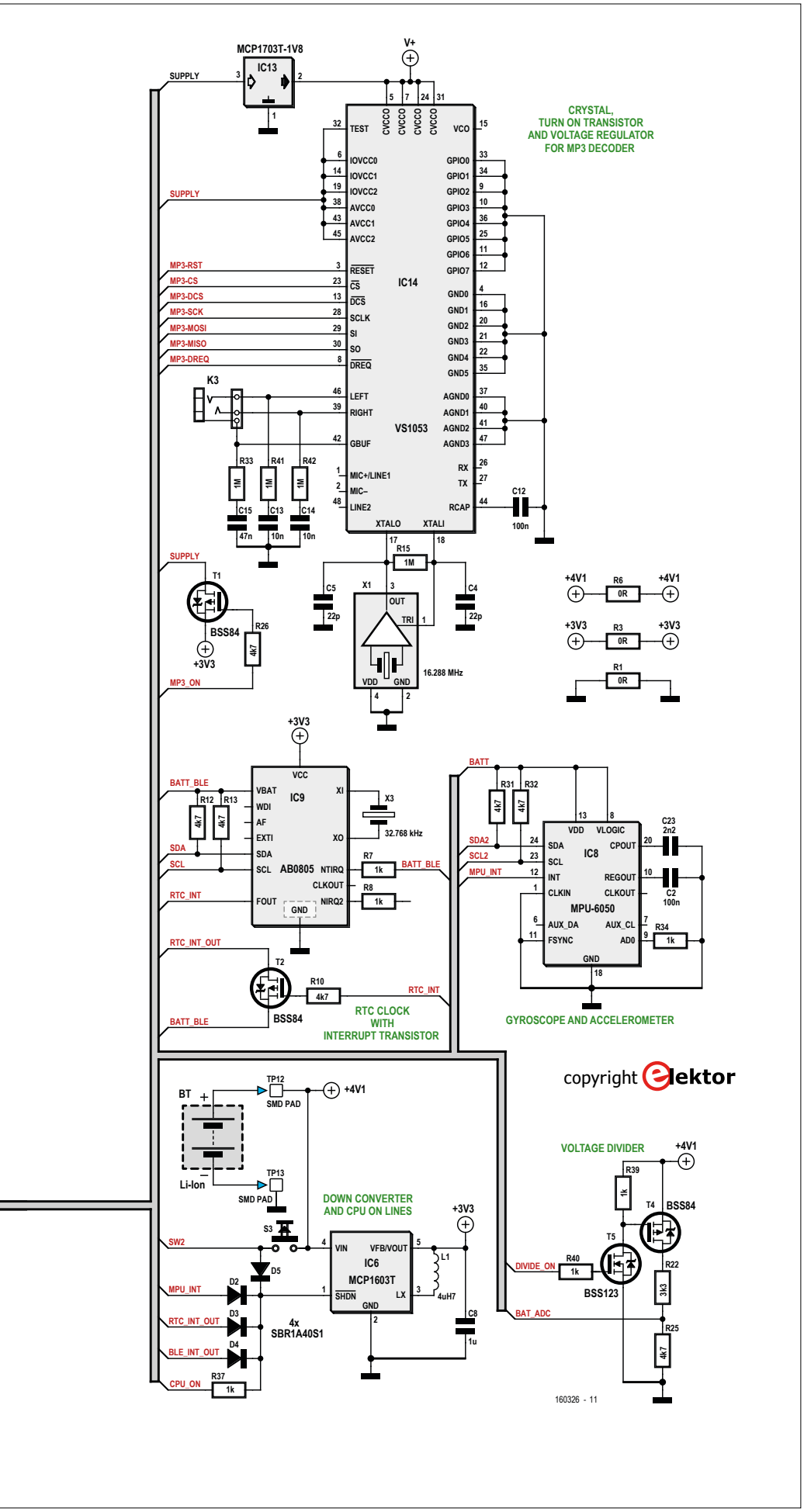


Bild 2. Im Schaltbild gibt es neben einigen Widerständen und Kondensatoren nur integrierte Schaltungen zu sehen.



eine Armbanduhr und groß genug für einen Touchscreen. Das Display erschien durchaus nicht zu kompliziert für ein Reverse Engineering - im Internet fand ich Artikel, in denen beschrieben wurde, wie man mit dem Display kommuniziert, und am Ende sogar ein Datenblatt des LCD-Controllers.

Trotz dieser überzeugenden Eigenschaften habe ich mich beim ersten Prototyp für eines dieser beliebten ILI9320-basierten TFT-LCDs entschieden. Damals dachte ich, dass die Anzeige viel schneller wäre, wenn ich statt eines 8-Bit- einen 16-Bit-Bus einsetzen würde. Außerdem kennt der Controller-Chip ILI9320 viel mehr integrierte Funktionen. Dieses Experiment war zwar interessant, aber nicht befriedigend. Das LCD hatte einen schlechten Betrachtungswinkel und die Farben waren recht blass. Vielleicht war mein TFT nicht von bester Qualität, aber dieses Experiment hat mich dazu veranlasst, zum Nokia-Display zurückzukehren. OK, ich weiß, es ist nicht rasend schnell, aber es ist gut genug für einfache Benutzeroberflächen. Denken Sie daran, dass die Art und Weise, wie die Grafik im Display gezeichnet wird, die Geschwindigkeit des Systems stark beeinflusst. Pixel für Pixel dauert es viel länger, als wenn man große Datenblöcke mit Direct Memory Access (DMA) überträgt. Das LCD wird an K4 angeschlossen.

Touchpanel hinzufügen

Das erste Panel, das ich ausprobiert hatte, stammte aus einem Nokia C3 von 2010. Es ist eine perfekte Ergänzung für das 2,2-Zoll-LCD des N82, ein etwas größerer und fast voll kompatibler Bruder des Nokia 6300. Obwohl resistiv, besaß es eine ziemlich gute Sensibilität. Leider war es zu groß für eine Armbanduhr und passte nicht zum 2“-Display des Nokia 6300. Nach vielen Stunden der Internetrecherche fand ich bei Aliexpress endlich ein geeignetes Touchpanel. Seine Größe war zwar perfekt für mein Display, aber die Sensibilität war

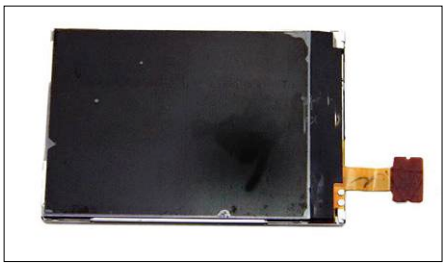


Bild 3. Das Nokia-Display 6300 ist noch im Internet verfügbar.

nicht beeindruckend: Man musste ziemlich kräftig mit dem Finger drücken und quetschen. Obwohl ich dieses Panel in der aktuellen nWatch verwende, bin ich nicht glücklich damit und möchte es demnächst durch ein kapazitives Touchpanel ersetzen. Die vier Touchpanel-Anschlüsse führen zu TP8...TP11.

Touchpanel-Controller

Die Steuerung des resistiven Touchpanels bleibt dem STMPE811 von STMicroelectronics (IC11) vorbehalten. Dieses speziell für diese Aufgabe geeignete IC kommuniziert mit dem zentralen Controller über SPI oder – wie hier – über I^2C. Es verfügt über viele Register, die eine tiefgreifende Steuerung des Touchpanels und eine Anpassung an das verwendete Display ermöglichen. Es hat auch einige interessante Hardware-Features wie Window-Tracking oder Temperaturmessung. Es kann auf eine Berührung hin einen Interrupt auslösen und damit die CPU veranlassen, die Koordinaten zu lesen (statt periodische Registerabfragen durchzuführen).

Hintergrundbeleuchtung

Der AP5724 im 6-Pad-DFN-Gehäuse (IC12) von Diodes Inc. ist ein kleiner Aufwärtswandler für weiße LEDs, der die Helligkeit des LCD steuert. Als Aufwärtswandler benötigt er einige externe Bauteile wie eine Leistungsinduktivität (L2), eine Schottky-Diode (D1) und zwei Kondensatoren (C17, C18). Die Helligkeitsinformation gelangt als PWM-Signal vom Controller zum Wandler. Er kann bis zu sechs LEDs ansteuern, und da das Nokia-Display nur zwei besitzt, erfüllt der AP5724 seinen Job perfekt. Der Widerstand R5 bestimmt den maximalen Strom durch die LEDs.

MP3-Player

In den ersten beiden Prototypen übernahm die Haupt-CPU das Abspielen von Musikdateien. Das digitale Signal wurde zur Ansteuerung der Ohrhörer von einem externen Digital-Analog-Wandler (DAC) analogisiert. Es wurden nur WAV-Dateien unterstützt, und alles - Datei lesen, Header-Parsing und so weiter – wurde von der Software erledigt. Da ich keinen blassen Schimmer hatte, wie MP3-Dateien in der Software zu entschlüsseln wären, suchte ich nach einer anderen Lösung. Ich entdeckte den exquisiten VS1053 von VLSI Solution (IC14), der verschiedene Dateiformate dekodieren und abspielen kann (Ogg Vorbis, MP3, AAC, WMA, FLAC, ja, sogar MIDI-Dateien). Alles, was man tun muss, ist, die Dateien über SPI zum Chip zu schicken. Dieser Chip erledigt nicht nur all die schwierige Dekodierarbeit, er bietet auch noch Funktionen zur Klang- und Lautstärkesteuerung. Zu allem Überfluss wird er noch von einem Kopfhörerverstärker abgerundet, kurzum, eine eierlegende Wollmilchsau! Dank IC14 ist alles, was die CPU tun muss, ab und an Datenblöcke an den Dekoder-Chip zu senden.

Der VS1053 ist recht klein und benötigt nur ein paar Kondensatoren, Widerstände und einen Quarz. Es begnügt sich mit 1,8 V für den Kern (C_{VDD}), was den Spannungsregler IC13 begründet. Der Dekoder kann mit Transistor T1 ein- und ausgeschaltet werden, um Strom zu sparen. **Bild 4** zeigt den MP3-Player auf der nWatch im Einsatz.

Bluetooth Low Energy (BLE)

Das Bluetooth-Modul (IC2) der nWatch basiert auf dem Bluetooth-Chip LE CC2540 von Texas Instruments und kann zur Kommunikation mit einem Smartphone verwendet werden. Hiermit können Benachrichtigungen angezeigt sowie die Uhrzeit und das Datum vom Handy aus eingestellt werden. Das Modul geht einen guten Kompromiss zwischen Preis und Platzbedarf ein. Die Reichweite hängt von vielen Faktoren ab und kann nicht zuletzt durch die Einstellung der Sendeleistung in den entsprechenden Registern geändert werden. Für eine kleine Leiterbahn-Antenne ist die Reichweite aber eigentlich ganz in Ordnung. Das Modul kann den Controller aus seinem Dämmerschlaf aufwecken, wenn Daten empfangen werden. Daten zwischen der CPU und dem BLE-Modul werden über eine simple serielle Verbindung übertragen.

Eine unangenehme Seite des Moduls bleibt doch: Es zieht signifikant viel Strom, wenn es aktiv ist. Deshalb ist es besser, es nach jedem Datenaustausch ab- und nur bei Bedarf wieder anzuschalten.

Sensoren

Die nWatch ist mit drei Positionssensoren ausgestattet, die unter Elektronikern sehr beliebt sind. Deshalb gibt es im Internet auch viele Open-Source-Bibliotheken. Der Sensorsatz besteht „traditionell" aus Gyroskop, Beschleunigungsmesser und Magnetometer.

Die ersten beiden Sensoren sind in IC8 kombiniert, dem MPU6050 von InvenSense, einem sechsachsigen Bauteil zur Bewegungsverfolgung. Das IC kommuniziert mit der CPU über einen I^2C-Bus. Es besitzt einen eigenen LDO-Spannungsregler, der es versorgt, auch wenn der Haupt-Gleichspannungswandler abgeschaltet ist. Das Interessante an diesem Chip ist der integrierte programmierbare digitale Bewegungsprozessor (DMP). Man kann dessen Firmware modifizieren und ihn eigene Berechnungen durchführen lassen, was die Haupt-CPU entlastet. Die Firmware wird kurz vor der Initialisierung auf den Sensor hochgeladen. Damit erhält der Sensor weitere Funktionen wie Klopf- und Freier-Fall-Erkennung. Bei solchen Ereignissen kann er Interrupts generieren, die die Haupt-CPU aus dem Schlafmodus holen.

Das 3-Achsen-Magnetometer ist der bekannte HMC5883 von Honeywell (IC7). Dieser kleine Chip im QFN-Gehäuse kommuniziert ebenfalls über einen I^2C-Bus, aber nicht über den gleichen wie IC8. Der Grund dafür ist einfach, dass die Leiterbahnen so kurz wie möglich gehalten und schwer zu lötende Vias vermieden werden sollten. Dies ist aber kein Problem, da die CPU zwei I^2C-Ports besitzt. Die Sensoren sind an den jeweils nächstliegenden I^2C-Bus angeschlossen.

Bild 4. Dank eines speziellen Dekoder-ICs kann die nWatch als MP3-Player verwendet werden.

Echtzeituhr

Damit die Zeitinformation erhalten bleibt, wurde eine Low-Power-Echtzeituhr (RTC) mit Alarmfunktionen in die Schaltung aufgenommen. Man muss so nur noch den Alarm der RTC entsprechend einrichten, bevor man die Uhr abschaltet. Sobald Datum und Zeit des Alarms erreicht sind, schaltet die RTC den Spannungsregler IC6 über den Transistor T2 ein und die Uhr wird wieder lebendig.

Ich habe eine gute RTC für meine Zwecke gefunden, die AB0805 von Abracom (IC9). Sie kommuniziert über eine I²C-Schnittstelle und verbraucht nur etwa 50 nA, viel weniger als die CPU im tiefsten Schlafmodus.

Stromversorgung

Die nWatch muss als tragbare Armbanduhr natürlich so wenig wie möglich Strom aufnehmen. Dies wird zunächst dadurch erreicht, dass der 3,3-V-Spannungsregler (IC6) über seinen Shutdown-Pin komplett ausgeschaltet wird. Es gibt vier Möglichkeiten, die Uhr wieder einzuschalten. Die offensichtlichste und einfachste zu verwirklichende Methode ist es, mit der Taste S3 eine positive Spannung über D5 an den SHDN-Pin zu legen, so dass der Regler sich und den Rest der nWatch einschaltet. D5 bildet mit D2, D3 und D4 ein Wired-Or-Gatter: Gyroskop/Beschleunigungsmesser (IC8), RTC (IC9) und das BLE-Modul (IC2) können den Schaltregler auf die gleiche Weise aktivieren.

IC6, ein MCP1603 von Microchip, ist ein synchroner Hochleistungs-Abwärtsregler, der 500 mA liefern kann. Er ist ideal für batteriebetriebene Anwendungen, nicht zuletzt, weil er wegen der hohen Schaltfrequenz von 2,0 MHz nur eine kleine Induktivität L1 (und zusätzlich auch nur ein einziges weiteres externes Bauteil, den Kondensator C8) benötigt.

Es gibt mit IC4 und IC5 zwei weitere LDO-Spannungsregler. Sie liefern Energie für die Teile der Uhr, die immer eingeschaltet sind (es sei denn, die Batterie ist erschöpft): IC2, IC8 und IC9. IC5 versorgt auch die CPU und verhindert, dass diese ihre batteriegepufferten Registerwerte verliert, wenn der 3,3-V-Regler abgeschaltet wird. Diese Register sind ein spezieller geschützter Bereich des CPU-Speichers, in dem Daten gehalten werden, solange die Spannung am VBAT-Pin hoch genug ist.

IC4 versorgt das BLE-Modul, so dass es den Rest der Schaltung nicht zu stark stört. Wie bereits erwähnt, liefert der LDO-Spannungsregler IC13 1,8 V für den Kern des MP3-Dekoders IC14. Über den Spannungsteiler R22 und R25 kann die CPU den Füllstand der Batterie messen. Dieser Teiler lässt sich durch T4 und T5 stromlos schalten, um Energie zu sparen.

LiPo-Batterielader

Energielieferant der nWatch ist eine 150-mAh-LiPo-Batterie. Sie ist sehr klein, kann aber die Uhr im Sleep-Modus für mehr als eine Woche versorgen. Wenn Sie nur ab und zu nach der Zeit sehen, reicht die Akkuladung für mindestens fünf Tage. Die Batterie ist mit einem Schutzmechanismus (protection circuit module, PCM) ausgestattet, der vor Kurzschlüssen und Tiefentladung schützt. Wenn die Batterie vollständig entladen ist, trennt das PCM die Batterie von allen Verbrauchern, bis ein Ladegerät angeschlossen wird. Dann allerdings ist der gesamte Speicher gelöscht, einschließlich der Datum- und Uhrzeitspeicher der RTC. Ich habe eine einfache App für mein Smartphone geschrieben, um die Wiederherstellung der Daten zu vereinfachen.

Für das Laden der Zelle ist der „Single Cell Charger" IC3 zuständig, ein MCP73831 von Microchip. IC3 benötigt nur einen externen Widerstand, um den gesamten Ladevorgang automatisch abzuhandeln.

Alle Verbinder

Die nWatch verfügt über drei Hauptverbinder: einen Steckplatz (K1) für die microSD-Karte auf der Unterseite, einen Mikro-USB-Anschluss (K2) und eine 3-polige Stiftleiste (TP5...TP7) zur Programmierung des Mikrocontrollers mit einem ST-Link-Programmer und/oder, um das laufende Programm zu debuggen. Dies ist die beste und schnellste Art, die Uhr zu programmieren.

Über den Mikro-USB-Anschluss werden sowohl die LiPo-Batterie aufgeladen als auch Daten ausgetauscht. Schließt man einen Computer an (Windows, Linux oder Mac OS), wird die nWatch als Massenspeichergerät erkannt und der Inhalt der SD-Karte angezeigt. Dank des integrierten Bootloaders kann man über den USB-Anschluss auch Programme auf die Uhr hochladen. Dies dauert jedoch viel länger als mit einem externen Programmer. Für die USB-Programmierung muss die ausführbare Datei (HEX oder BIN) zudem zuerst in ein spezielles DFU-Format (Device Firmware Update) konvertiert werden. Auch ist ein Debuggen irgendwelcher Art über USB nicht möglich. Deshalb sollte man die USB-Programmierung eher als ein zusätzliches Feature ansehen als einen echten Software-Entwicklungsport.

Die Platine

Die Form der Platine ist an das 3D-gedruckte Gehäuse angepasst und weist zudem eine Aussparung für die 3,5-mm-Klinkenbuchse auf. Indem man die Buchse „in" statt auf der Platine montiert, spart man eine Menge Platz und kann das Uhrengehäuse schön dünn halten.

Auch wenn die Leiterbahnen bisweilen nur 6 mil breit sind, kann man die Platine in Heimarbeit auch im Foto-Positiv-Verfahren anfertigen. Dies ist auch für die „wahren" Heimwerker eine echte Herausforderung, da das Verlöten der ungefähr 200 Vias von Hand nichts für schwache Nerven respektive zittrige Finger ist.

Der Mikrocontroller in seinem großen, 144-poligen LQFP nimmt einen Großteil der Platinenfläche in Anspruch. Auf dem Board geht es deshalb ziemlich eng zu, ich glaube nicht, dass ich es hätte noch kleiner machen können. Ich habe deshalb kurz mit einer vier- oder sechslagigen Platine und mit einem Controller im BGA-Gewand geliebäugelt, aber diese Idee wieder fallengelassen, weil die nWatch ja ein kostengünstiges Entwicklungsboard für Bastler werden sollte und nicht nur ein teurer Prototyp. Kleinere Bauteile würden zudem die Bestückung verkomplizieren, da man BGA-Bauteile nicht mal so eben mit dem Lötkolben auf die Platine bringen kann. Die Positionierung der Bauteile auf der Platine ist von entscheidender Bedeutung, was die endgültige Bauhöhe der Uhr angeht. Ich habe alle Teile mit einer nennenswerten Höhe (Buchsen, Schalter) auf der gleichen Platinenseite angeordnet. Die LiPo-Batterie ist problematisch. Eigentlich wäre es am besten, sie in einem gefrästen Schlitz in der Platine unterzubringen. Alternativ kann man auf der Platine einen Bereich mit ausschließlich flachen Bauteilen (zum Beispiel dem nur 1 mm dicken SRAM) schaffen und die Batterie darüber platzieren (**Bild 5**). Andere Elemente wie der 3,5-mm-Klinkenstecker oder die USB-Buchse können „schwebend" in Aussparungen der Platine montiert werden. Für alle, die sich nicht mit solchen Feinheiten der Platinenherstellung abgeben möchten oder sich die Anfertigung einer solchen Platine nicht zutrauen, haben wir im Elektor-Labor ein Layout vorgefertigt (**Bild 6**), das Sie von der Projektseite [1] herunterladen können. Oder Sie machen es sich ganz einfach und bestellen eine fertige Leerplatine im Elektor-Shop!

Teile im 3D-Druck

Das Schwierige am 3D-Druck ist nicht der Druck selbst, sondern die Erstellung der 3D-Modelle der Teile. Das Ganze muss nicht nur nett aussehen, auch die Pla-

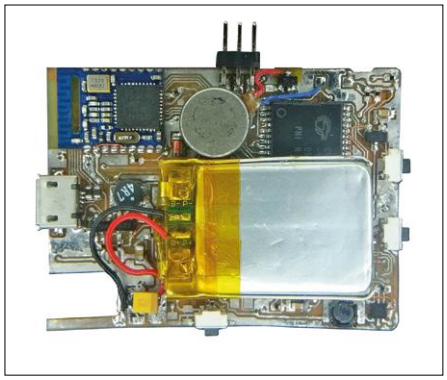


Bild 5. Die LiPo-Batterie wird auf den SRAM-Chip gelegt.

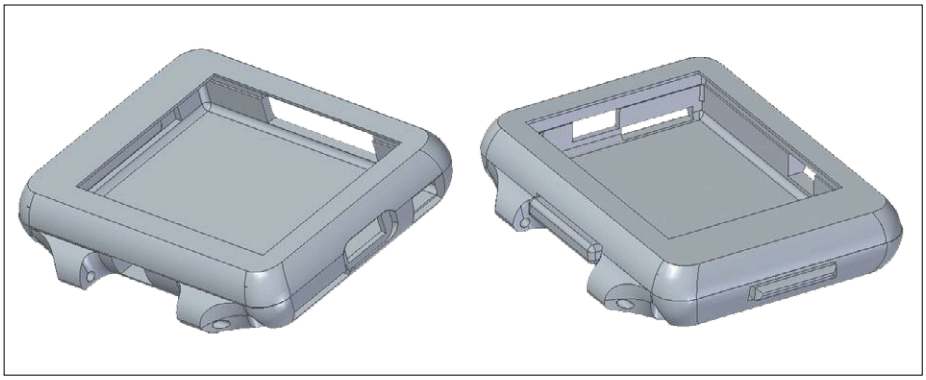


Bild 6. 3D-Zeichnungen des Uhrengehäuses.

tine mit all ihren Teilen muss hineinpassen. Die Entscheidung, eine Uhr zu bauen, macht es nicht leichter, da alles sehr klein und sehr dünn/fragil ist. Man zeichnet zunächst eine grobe Form der Uhr unter Berücksichtigung der Platinengröße, der LiPo-Batterie und des LCDs mit Touchpanel.

Leider habe ich keine einfachere Lösung für dieses Problem entdeckt, als alles aufs Genaueste zu vermessen, eine erste Probeversion zu drucken und zu versuchen, die Uhr zusammenzubauen. Nur so lassen sich alle Fehler feststellen. Das Modell wird entsprechend geändert und gedruckt und dann wird erneut versucht, die Uhr zusammenzusetzen. Im dritten oder vierten Anlauf sollte es klappen!

Die Teile, aus denen sich das nWatch-Gehäuse zusammensetzt, wurden im 3D-CAD-Programm Solid Edge von Siemens (**Bild 7**) entworfen. Es gibt eine kostenlose Testversion, die aber nur 45 Tage genutzt werden kann. Ich hatte zunächst die Idee, ein zumindest spritzwassergeschütztes Gehäuse zu entwerfen, aber vor allem die Abdichtung der externen Anschlüsse (vor allem des SD-Kartensteckplatzes) stellte sich als eine nicht zu überwindende Hürde heraus. So musste ich schweren Herzens beschließen, ein zweiteiliges Gehäuse zu entwerfen, das von vier Schräubchen zusammengehalten wird. Dies ist eine gute Lösung, weil man nach dem Entfernen der Schrauben einen einfachen Zugang zur Hauptplatine, zur LiPo-Batterie und zum LCD mit seinem Touchpanel hat.

Das Uhrengehäuse hat zwei rechteckige Schlitze für die Tasten. Die Wake-up-Taste über dem Uhrenarmband ist die größere, damit man sie bequem drücken kann. Übrigens: Auch die Tasten werden 3D-gedruckt. Alle Dateien zum 3D-Druck des Uhrengehäuses finden sich ebenfalls im Download-Paket auf der Projektseite [1].

Die Uhr sieht am besten aus, wenn sie mit einer sehr niedrigen Schichthöhe gedruckt wird. Mit einer Z-Achsen-Schrittweite von 0,2 mm habe ich zufriedenstellende Ergebnisse erzielt. Mein 3D-Drucker *Prusa Mendel Iteration 2* (I2) erwies sich in punkto Auflösung und Stabilität als gut genug, um ein ziemlich schönes Gehäuse zu drucken. Das zeigt, dass auch selbstgedruckte/selbstgebaute und preisgünstige 3D-Drucker einen solchen Job ganz gut erledigen können.

So setzen Sie ihre nWatch zusammen

1. Alle Teile organisieren

Aller Anfang beginnt mit dem Organisieren der Bauteile, vorzugsweise mit den problematischen zuerst. Dann weiß man direkt, was man sich mit dem Entschluss, die nWatch zu bauen, eingebrockt hat. Das Touchpanel ist wahrscheinlich am schwierigsten zu besorgen, aber auch die „mittenmontierbare“ 3,5-mm-Audio-Klinkenbuchse ist so ein Teil. Ich habe sie aus einem alten Smartphone ausgeschlachtet und in meinem Prototypen eingesetzt. Leider habe ich so keinen Hersteller oder Produktnummer erfahren, eine Internet-Recherche blieb erfolglos. Im Internet erhältliche Buchsen hatten entweder einen anderen Footprint, eine inkompatible Kunststoffabdeckung oder beides. Es ist jedoch eventuell möglich, die Aussparung in der Platine mit einer kleinen Feile anzupassen oder die Buchse mit Drahtbrückchen an den Pads anzuschließen.

Die Drucktasten sind weniger problematisch, können aber auch schwer zu besorgen sein. Ich habe keine genaue Referenz für sie, nur die Abmessungen. Schauen Sie sich Drucktaster im Internet an und

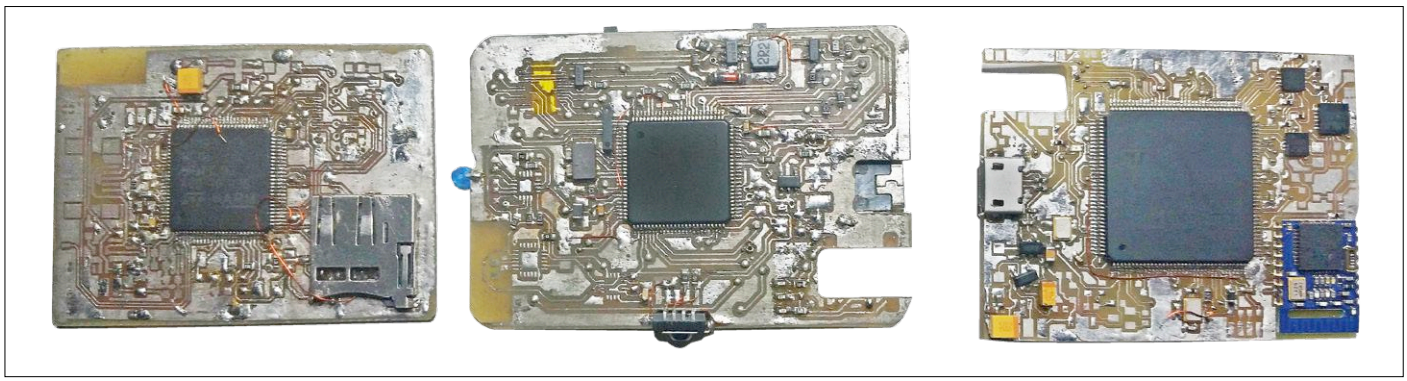


Bild 7. Die Platine musste mehrmals umgestaltet werden, bevor das endgültige Design erreicht war.

vergleichen Sie ihre Abmessungen: Dies scheint immer noch der beste Weg, um an passende Drucktaster zu kommen.

2. Bestücken Sie die Platine

Wenn Sie alle Bauteile zusammen haben und die nackte Platine auf Ihrem Labortisch liegt, können Sie die Bestückungsarbeiten beginnen (**Bild 8**). Das Löten der meisten Teile auf der Platine sollte nicht so kompliziert sein, wenn Sie einen kleinen Lötkolben oder besser eine Lötnadel verwenden. Die kleinsten Teile sind die Widerstände und Kondensatoren im 0402-Format, für die man Erfahrung im SMD-Löten mitbringen sollte. Beginnen sollte man mit der Montage der SMD-Quarze und der Chips in den QFN-Gehäusen mit ihren Anschlüssen auf der Unterseite des Gehäuses. Dazu ist eine Heißluftstation erforderlich. Auch wenn Sie zum ersten Mal mit QFN-Chips zu tun haben: Fürchten Sie sich nicht! Wenn Sie alles gut vorbereiten, muss nur noch der Chip platziert und erhitzt werden. Alles andere geht dann wie von selbst. Setzen Sie das Bauteil aber nicht zu lange der heißen Luft aus, denn auch heiße Luft kann für ein Bauteil tödlich sein! Der Rest ist dann nichts anderes als „normales" SMD-Löten. Achten Sie darauf, dass bei den größeren ICs (Controller, VS1053 und so weiter) wirklich alle Pins korrekt verlötet sind. Manchmal sieht es so aus, als ob, aber sie hängen doch ein wenig über der Leiterplattenoberfläche, ohne das Pad zu kontaktieren.

3. Setzen Sie alles zusammen

Nach der Bestückung der Platine sollten Sie genauestens überprüfen, ob es keine Kurzschlüsse oder andere Lötfehler gibt. Erst wenn Sie absolut sicher sind, dass alle Lötverbindungen korrekt sind, dürfen Sie die Batterie anschließen und auf der Platine oberhalb des SRAMs (IC10) platzieren. Mein Touchpanel hatte bei der Lieferung schon Klebeband an den Ecken, so dass ich nur den Schutzfilm entfernen und es auf das gesäuberte LCD kleben musste. Was aber Probleme verursachen kann, sind die unterschiedlichen Anschlussbelegungen von verschiedenen Panelversionen. Man kann die Reihenfolge der vier Pins nur raten. In meinem Fall musste ich zwei Drähte kreuzen, damit es richtig funktioniert. Zum Glück geht aber nichts kaputt, wenn man die Drähte vertauscht! Wenn die Koordinaten vertauscht sind oder Berührungen nicht erkannt werden, tauschen Sie einfach die Drähte solange, bis alles wie erwartet funktioniert (**Bild 9**).

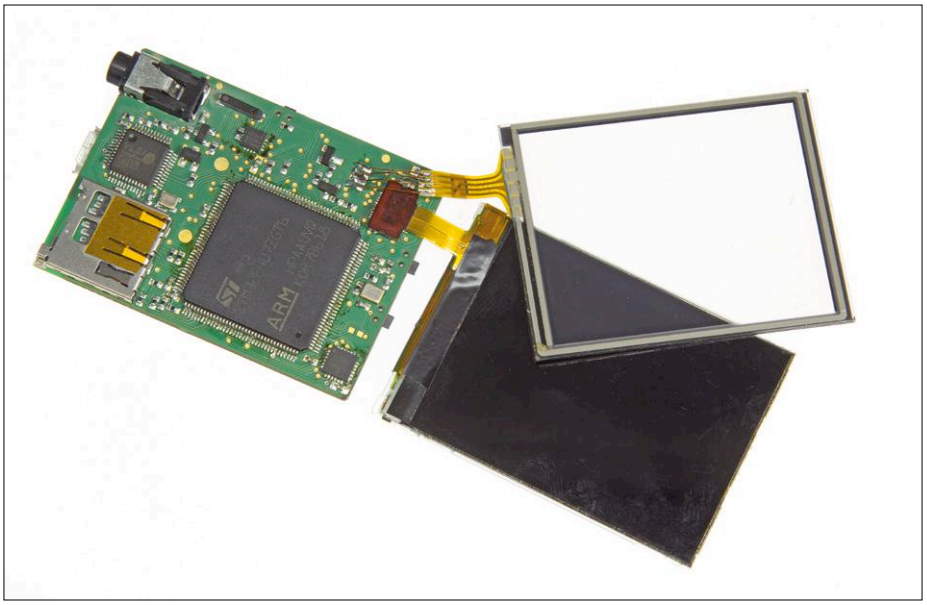


Bild 8. Zusammengebauter Prototyp mit Display und Touchscreen.

Nachdem die Panelanschlüsse korrekt auf die Hauptplatine gelötet wurden, kann man den LCD-Verbinder einstecken und das Display auf die andere Seite der Platine drehen, wo es die Batterie verdeckt. Die Gehäusemontage ist wirklich einfach (**Bild 10**). Mit einem Stück feinen Sandpapiers schleifen Sie die Flächen, wo sich die beiden Gehäuseteile berühren, eben und glatt. Dies ist wichtig für eine perfekte Passform. Auch sollte man die Löcher im Unterteil und im Halter des Uhrbandes vorsichtig nachbohren, denn nach dem Druck sind sie meist ein wenig mit Filament verschmutzt. Dann legen Sie die seitliche Taste an ihren Platz, gefolgt von der Platine (denken Sie daran, die microSD-Karte vorher zu entfernen). Legen Sie den Wake-up-Knopf in den Schlitz und decken Sie alles mit dem oberen Teil des Gehäuses ab. Nun die vier Schrauben einsetzen und fertig ist die nWatch.

Nein, nur beinahe. Sie wollen die Uhr ja am Arm tragen. Im Netz gibt es viele Seiten, die eine Unmenge der verschiedensten Armbänder samt Federstangen anpreisen, die passen (Stegbreite beachten!) und die man auch ohne Spezialwerkzeug montieren kann.

Software

Während ich entwickelte und verschiedene Hardwareversionen aufbaute, schrieb ich

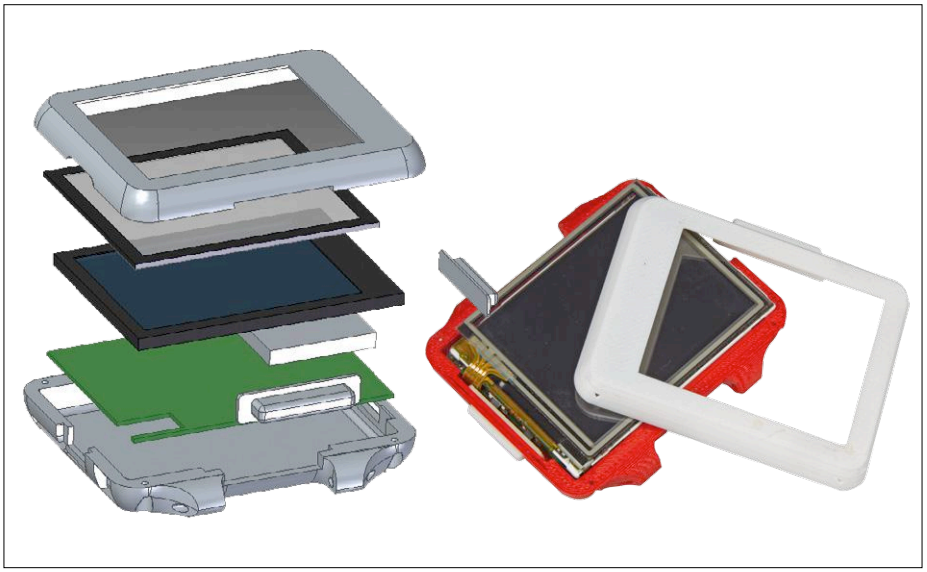


Bild 9. Die nWatch explodiert: So stapeln Sie die Teile für die Endmontage.

STÜCKLISTE

Widerstände

R1,R3,R6,R9 = 0 Ω, 0402
R2,R5 = 10 Ω, 0603
R4,R7,R8,R14,R16-R21,R27...
R30,R35,R36,R42,R46...R48,R55 = 1 k, 0402
R10...R13,R31,R32,R63,R66 = 4k7, 0402
R15,R49-R51,R85 = 1 M, 0402
R54 = 5k6, 0402
R67 = 3k3, 0402

Kondensatoren

C1,C6 = 1 µ, 0402
C2,C7,C16,C18...C21,C39 = 100 n, 0402
C3 = 10 µ, 0805
C8,C9,C17,C25 = 1 µ, 0805
C10,C11,C40,C41 = 22 p, 0402
C26 = 4µ7, 0603
C29 = 2n2, 0603
C35 = 47 n, 0402
C36,C38 = 10 n, 0402

Induktivitäten

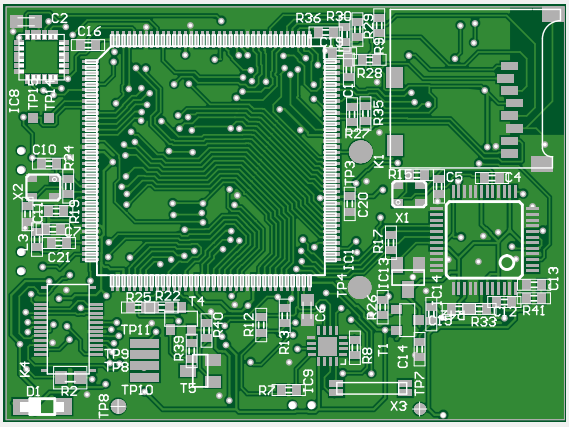
L1 = 4µ7, Murata LQM31PN4R7M00L, 1206
L2 = 22 µ, Murata LQH31CN220K03L, 1206
L3...L5 = 10 µ, 0402

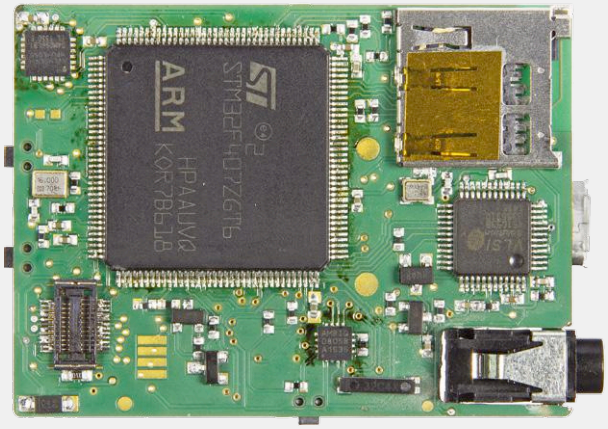
Halbleiter

D1-D5 = SBR1A40S1
IC1 = STM32F407ZGT6
IC2 = RF-BM-S02
IC3 = MCP73831
IC4, IC5, IC15 = MCP1703T-3302E/CB
IC6 = MCP1603T-330I/OS
IC7 = HMC5883L
IC8 = MPU-6050
IC9 = AB0805
IC10 = CY7C1041DV33-10ZSXI
IC11 = STMPE811
IC12 = AP5724
IC14 = VS1053
LED1 = LED, 0603
T1-T4 = BSS84
T5 = BSS123

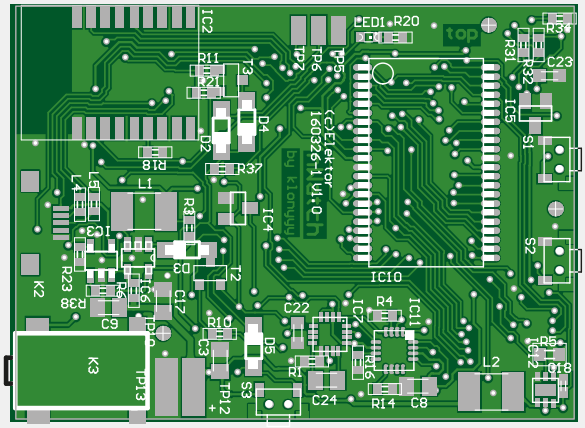
Außerdem

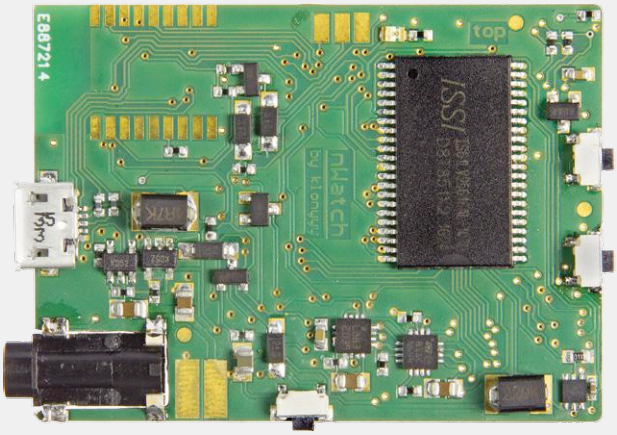
K1 = MicroSD-Kartenbuchse MEM2061-01-188-00-A.
K2 = Micro-USB-Type-B-Verbinder
K3 = Stereo-Klinkenbuchse 3,5 mm
K4 = LCD Nokia 6300 (oder Äquivalent) + Verbinder
S1-S3 = Taster, seitliche Montage, 3,45×2,2×4mm, TACTDR345H40B160
X1 = Quarzoszillator 12,288 MHz, SG-210STF
X2 = Quarz 16 MHz, 3,2×2,5 mm
X5 = Uhrenquarz 32,768 kHz, MC-146
Resistives Touchpad 2”
3D-gedrucktes Uhrengehäuse
Uhrarmband + 2 Federstege
LiPo-Batterie 150 mAh
Platine 160326-1





150% vergrößert





auch einiges an Code, um die Uhr zu steuern und um die vielen Bibliotheken für den Einsatz in der Uhr anzupassen.

Benutzeroberfläche mit STemWin

Eine der wichtigsten Bibliotheken für die nWatch ist STemWin. Dies ist eine Grafikbibliothek, die das LCD steuert und alle Elemente der Benutzeroberfläche zeichnet. Sie basiert auf emWin von Segger und wird von STMicroelectronics, dem Hersteller des Hauptcontrollers, zur Verfügung gestellt. Logisch, dass sich die Software ganz einfach an jedes STM32-IC anpassen lässt. Im Internet finden Sie viele verschiedene Beispiele und eine vollständige Dokumentation.

Wenn man die Beispielprogramme betrachtet, erinnern die Bildschirmelemente (Schaltflächen, Textfelder) an alte Windows-Zeiten. Allerdings und glücklicherweise können diese Teile der Schnittstelle nach Belieben geändert werden, was auch der Grund ist, warum ich diese Bibliothek für so interessant halte. Zusammen mit der Bibliothek erhält man mehrere Utilities, die bei der Zusammenstellung des Interfaces sehr nützlich sind. Mit dem GUIBuilder, meiner Meinung nach das nützlichste Werkzeug von allen, lässt sich sehr einfach ein interaktiver Screen entwerfen. Der automatisch generierte Code wird anschließend nur in das Projekt eingefügt. Dieses intuitive Werkzeug ist ein echtes Zeitsparwunder!

Multitasking mit FreeRTOS

Die Uhr wird von dem kostenlosen Echtzeit-Betriebssystem FreeRTOS gesteuert, das Hobbyisten wie Profis gleichermaßen zu schätzen wissen. Dank der Multitasking-Fähigkeit können mehrere Anwendungen gleichzeitig laufen. Man kann solchen Tasks Prioritäten verleihen, andere „schlafen legen“, sie eine bestimmte Zeit arbeiten lassen und vieles mehr. Ich will hier nicht ins Detail gehen, denn die Software ist wirklich gut dokumentiert. Konsultieren Sie bitte das Internet, wenn Sie mehr darüber erfahren möchten.

Animationen abspielen mit libjpeg

Eine andere bekannte Bibliothek, die ich verwende, ist libjpeg, die JPEG-Dateien dekodieren kann. Ich habe sie genutzt, um einfache Filme auf der Uhr abzuspielen. Der Hauptvorteil der Bibliothek ist ihre Geschwindigkeit, die mir erlaubte, bis zu 21 Frames pro Sekunde (FPS) mit 240x320 Pixeln darzustellen. Allerdings belegt die Bibliothek eine Menge an Stack-Speicher, weil sie einst entwickelt wurde, um die Verbreitung des JPEG-Standards zu fördern. Wenn Sie eine wirklich schnelle JPEG-Dekodierung benötigen und Sie sich keine Gedanken um den Speicherplatz machen müssen, ist die Bibliothek aber genau richtig.

Grafikbibliothek mit geringem Speicherbedarf

In Fällen, in denen der Speicherplatz kritisch ist, sollte man besser beispielsweise die „Tiny JPEG Decompressor“-Bibliothek TJpgDec verwenden. Diese kleine Bibliothek wurde von ChaN geschaffen, einem für seine FatFS- und Petit-FatFs-Bibliotheken zur Verwendung von SD-Karten in Embedded-Anwendungen bekannten japanischen Entwickler. Sie ist optimal für kleine, eingebettete Systeme. Da TJpegDec für kleine Mikrocontroller-Systeme konzipiert wurde, benötigt sie nicht so viel Speicher wie libjpeg. Und sie ist schnell genug, um zum Beispiel ein Foto zu machen. Ein Video abzuspielen liegt allerdings außerhalb der Möglichkeiten der Bibliothek.

Hardwarekonfiguration und ST-Bibliotheken

Die Low-Level-Bibliotheken von ST, obwohl sehr umfassend, haben keinen guten Ruf. Einige Anwender glauben, dass sie ineffizient sind und verwenden nur Bare-Metal-Programmierung, andere wiederum nutzen nur die Standard-Peripherie-Bibliothek. Persönlich hatte ich aber keine Probleme, diese Bibliotheken anzuwenden, obwohl ich manchmal auch ein Hardware-Register selbst konfigurieren musste, zum Beispiel im Fall einer periodisch aufgerufenen Funktion, die (wie ein DMA-Transfer) schnell ausgeführt werden muss. Wenn Sie die neueste Hardware-Abstraktionslayer-Bibliothek (HAL) von ST verwenden, müssen Sie in meinem Code alle benötigten Header-Dateien und mehr hinzuzufügen. Einige Leute haben mit dem STM32CubeMX-Initialisierungscode-Generator HAL-Funktionen erstellt, aber ich habe damit keine Erfahrung.

Über den Autor

Piotrek ist ein 19-jähriger Gymnasiast aus Polen. Er ist Fan eingebetteter Systeme und liebt die Mikrocontroller-Programmierung, vor allem von AVR- und STM32-Bausteinen. Seine Elektronik-Karriere begann mit dem Aufbau von Sumo-Robotern für Wettbewerbe, und kürzlich entdeckte er die Technik von Wearables für sich. In seiner knappen freien Zeit baut er auch ferngesteuerte Flugmodelle.

Fast Forward Award 2016

Die nWatch war einer der Beiträge beim Fast Forward Award, den Elektor gemeinsam mit den Veranstaltern der electronica 2016 in München ins Leben rief. Der Wettbewerb war eine großartige Gelegenheit für Profis und für Hobbyisten, Ideen, Projekte, Produkte und Startups zu präsentieren.

Der Autor dieses Artikels präsentierte seine nWatch einem internationalen Fachpublikum und einer Jury. So hatte er Gelegenheit, mehrere Hersteller zu treffen, um über die Bauteile der Uhr zu diskutieren und Anregungen von vielen anderen Teilnehmern und Besuchern zu erhalten. Mit dem Kopf voller neuer Ideen und dem dritten Preis in der FFA-Prototypen-Kategorie konnte Piotrek sich glücklich auf den Heimweg machen.

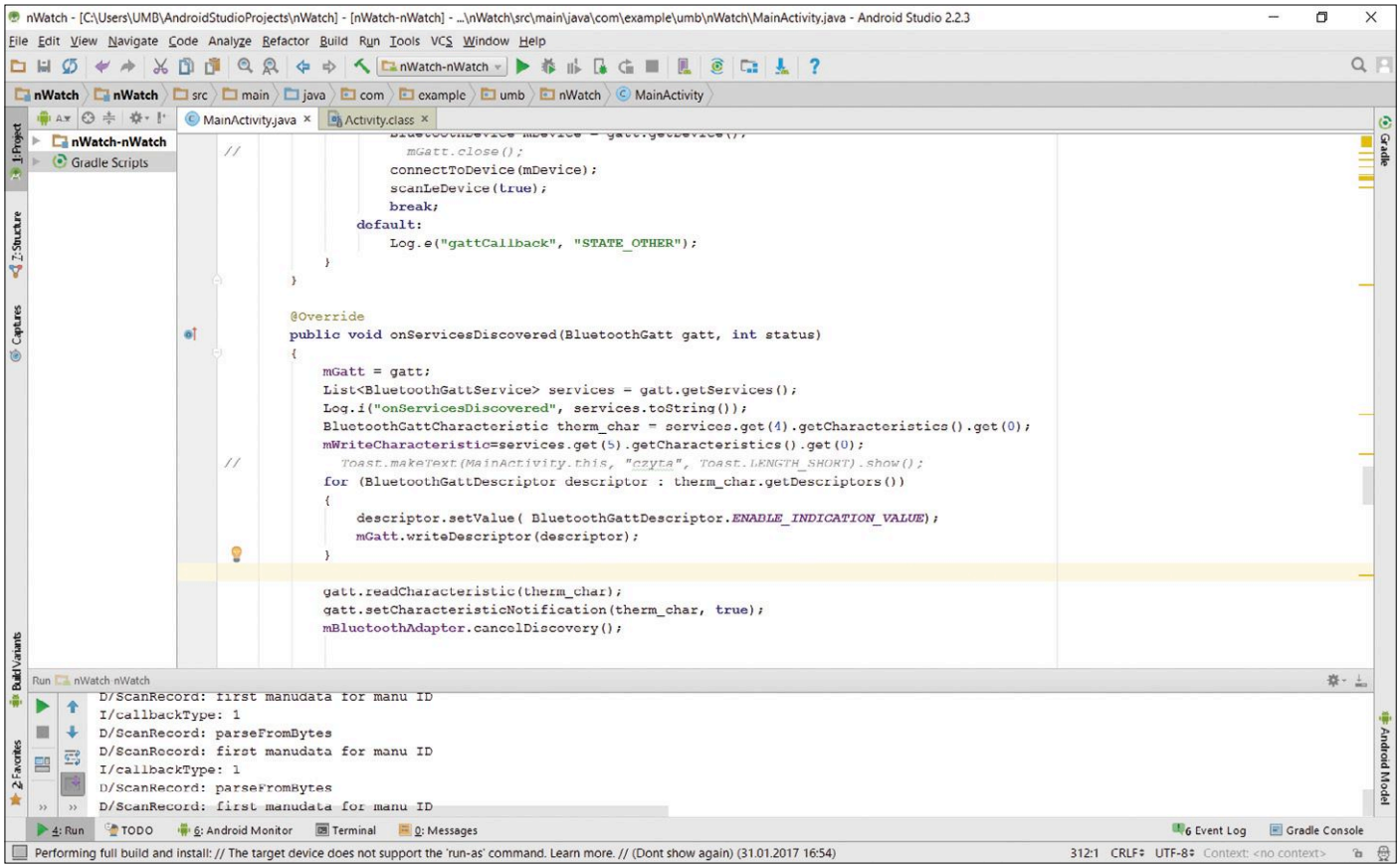


Bild 10. Die einfache Android-App, über die man mit der Uhr kommunizieren kann, wurde mit dem Android-Studio 2.2.3 erstellt.

Integrierte Entwicklungsumgebung

Die integrierte Entwicklungsumgebung (IDE), die ich ursprünglich zum Schreiben der nWatch-Software verwendet habe, war CoIDE von CooCox, weil ich eine unbegrenzte Code-Größe und eine Eclipse-ähnliche Schnittstelle wollte. Ich kann mich nicht über diese IDE beschweren, aber sie scheint von ihren Schöpfern recht stiefmütterlich behandelt zu werden. Die aktuelle Version ist ziemlich fehlerbehaftet, so dass ich mit einer älteren Version arbeien musste. Zu guter Letzt bis ich dann zu Attolics TrueSTUDIO gewechselt. Diese IDE wird laufend verbessert und gut unterstützt. Es gibt aber noch etliche Programmierumgebungen, die man verwenden kann (einschließlich einer „blanken" Eclipse mit Toolchain), auch wenn sie manchmal Codegrößenbeschränkungen haben oder nur als Testversionen verfügbar sind. Die komplette Konfiguration von Eclipse für die nWatch könnte Ihnen aber Kopfschmerzen bereiten und Sie den Mut verlieren lassen.

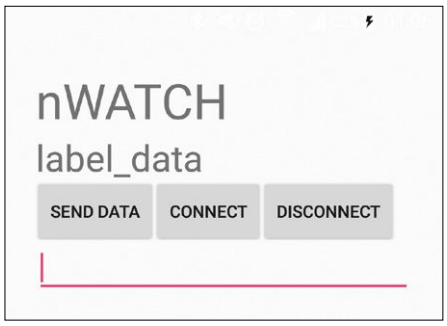


Bild 11. Die Android-App kann die Uhrzeit und das Datum der nWatch durch einen Tipp auf die „Send Data"-Taste einstellen.

In-circuit-Programmierung

Ich empfehle Ihnen dringend, in einen billigen ST-Link-Programmer zu investieren. Sie werden sehen, dass die Arbeit damit im Vergleich zum USB-Bootloader nicht nur viel bequemer ist, Sie können damit auch ihren Code debuggen. Der Programmer ist auch Teil der STM-Discovery-Kits und der Nucleo Boards (die oft auf Messen verschenkt werden). Sie müssen nur noch ein Kabel mit einer 3-poligen 2-mm-Buchse anfertigen, damit Sie loslegen können.

Android-App

Obwohl ich nicht gerade Java-Programmierer bin, wollte ich doch eine einfache App schreiben, um von einem Smartphone aus Daten an die Uhr zu senden. Ich habe ein Beispiel für eine Bluetooth-LE-App gefunden und sie modifiziert, damit ich die Uhrzeit und das Datum vom Telefon ablesen und drahtlos an die Uhr senden kann (**Bild 11**). Das war wirklich bequem, denn so war es nicht notwendig, eine spezielle App für die Uhr zu schreiben. Wenn die App auf dem Smartphone startet, sucht sie nach einer nWatch in der Nähe, und wenn sie eine gefunden hat, tippt man einfach auf die Sende-Taste, um die Uhrzeit und das Datum einzustellen. Die nächste Version der App soll die Uhr dann völlig ohne Benutzereingriff aktualisieren. ⏮

(160326)

Weblink

[1] www.elektormagazine.de/160326

HomeLab Helikopter

Zusammengestellt von **Clemens Valens** (Elektor Labs)

Arduino Create

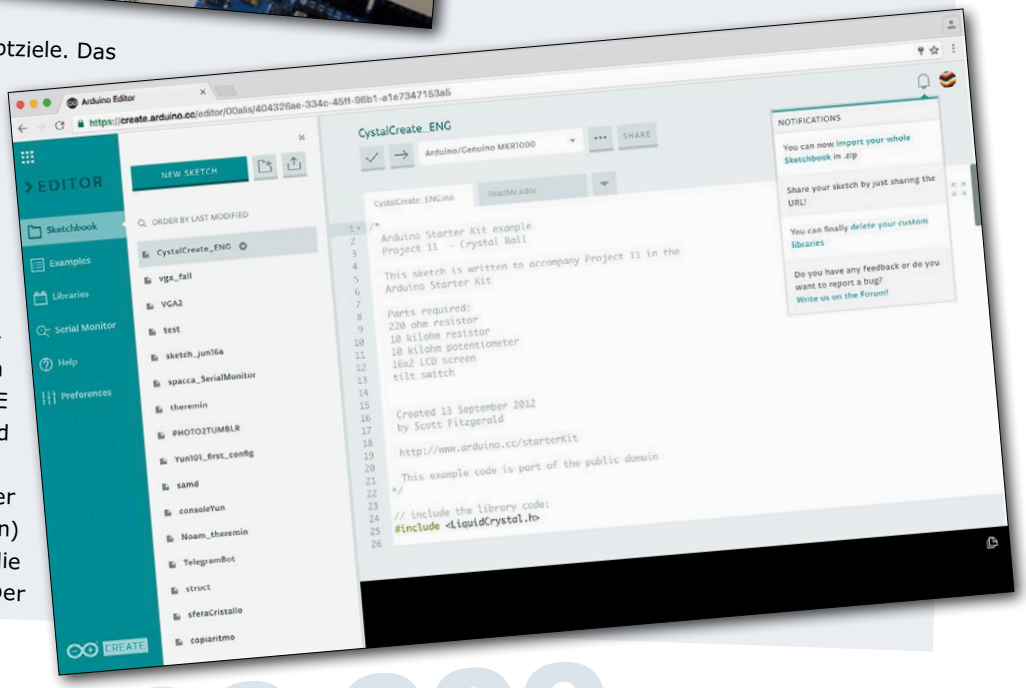
Obwohl sie noch nicht viel Staub aufgewirbelt hat, entwickelt sich die neue Arduino-Plattform spektakulär. In den vergangenen Monaten haben die Leute von Arduino still und heimlich an einer vollwertigen Cloud-basierten Online-Entwicklungsplattform namens *Arduino Create* gebastelt. Es ist ein ehrgeiziges Projekt, das noch einen Schritt weiter geht als das Konzept des jüngst verschiedenen Codebenders, bis hoch auf das Niveau der vorgefertigten BBC-micro:bit-Lösung. Die regelmäßigen Besucher der Arduino.cc-Website kennen diese Plattform schon von ihren Anfängen an, aber explizit beworben wurde ihr Vorhandensein nicht gerade.

Arduino Create hat derzeit zwei Hauptziele. Das erste ist es, den Stress bei der Installation von Boards, Tools und Bibliotheken völlig zu vermeiden. Der Anwender von Arduino Create meldet sich über sein Konto an, verbindet ein kompatibles Board mit dem Computer und startet die Codierung – so einfach sollte es sein. Das Board wird von der Online-IDE erkannt, alle im Arduino-Repository veröffentlichten Bibliotheken sind ohne Installation verfügbar, die IDE ist immer auf dem neuesten Stand und so weiter.

Das ähnelt sehr mbed von ARM (aber verzichtet auf den Programmierer-Jargon) oder der BBC-micro:bit-Umgebung (die aber nur dieses eine Board kennt). Der wichtigste Unterschied zu diesen Plattformen ist, dass Arduino Create kann, was Codebender auch konnte, nämlich das Board selbst programmieren, ohne dass der User manuell eine Datei auf einen Datenträger kopieren muss. Dazu muss zuerst eine kleine Software, die dem Browser Zugriff auf das Board erlaubt, installiert werden. Also, es ist nicht völlig unproblematisch, aber sobald dies erledigt ist, geht alles andere wie von selbst. Fans und Kenner der traditionellen Arduino-IDE können weiterhin ihre Off-line-Entwicklungsumgebung nutzen. Allerdings werden sie beim Herunterladen einer neuen Version von der Website feststellen, dass Arduino Create als der nun zu bevorzugende Weg dringend empfohlen wird.

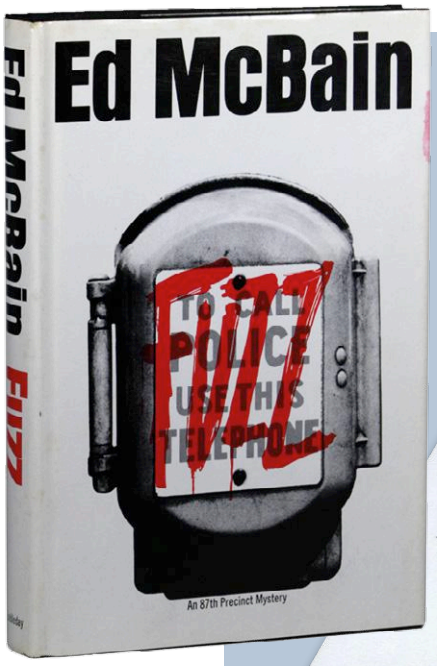
Das zweite Ziel ist es, die Entwicklung von IoT-Anwendungen so einfach wie möglich zu gestalten, indem kostengünstige Cloud-Services für Einsteiger bereitgestellt werden. Auch dies ist sehr ähnlich wie bei mbed, aber auch für Nicht-Spezialisten angenehm einfach. Schauen wir, was die Zukunft bringt. Arduino OS vielleicht?

(160404-a)



8.400.000.000

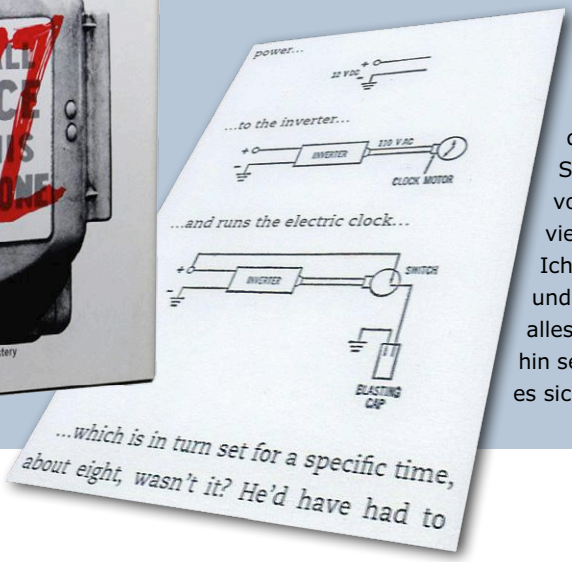
Im Jahr 2017 gibt es 8,4 Milliarden vernetzte Objekte.



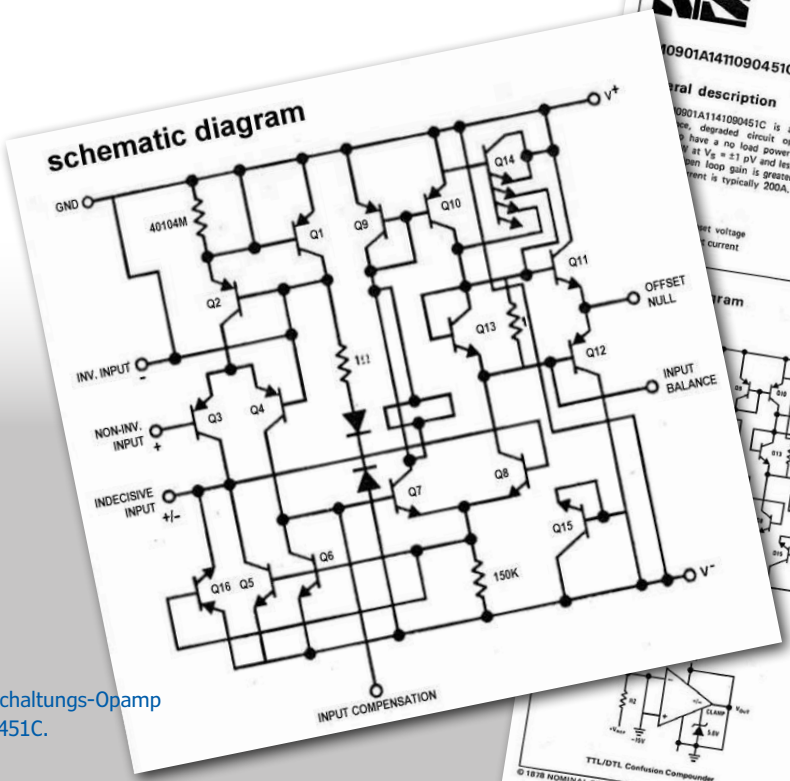
Elektronik in der Literatur

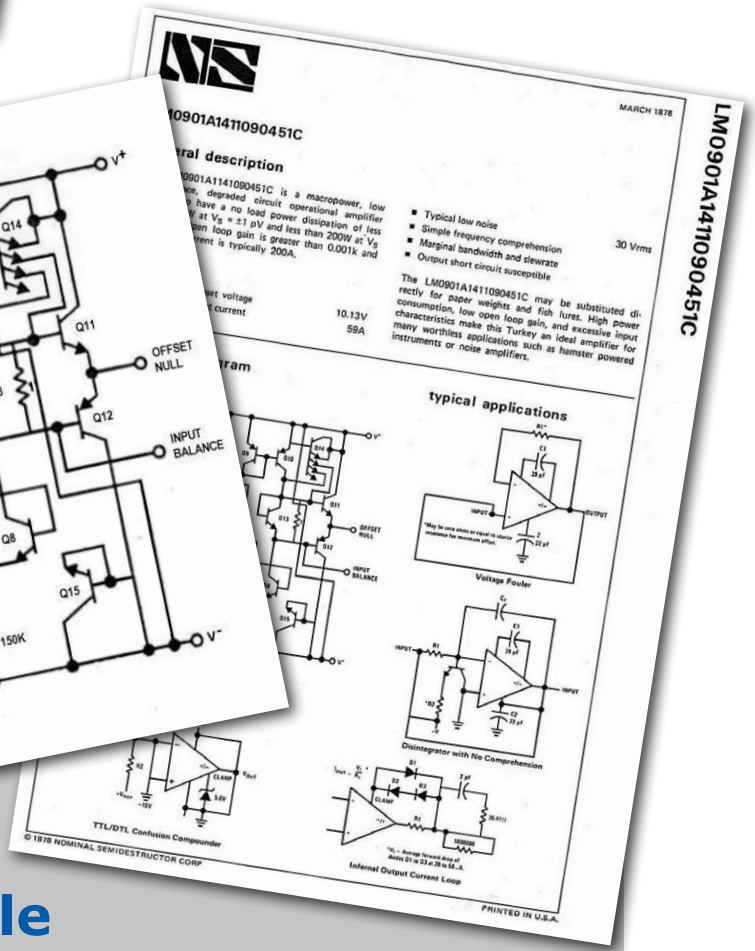
Soweit ich weiß - aber was weiß ich? – verstehen sich Literatur und Elektronik nicht so gut. Ok, es gab natürlich 1886 den Darlington-Fälschungsskandal, der von Sherlock Holmes aufgeklärt wurde, aber es ist seit diesem Vorfall viel Wasser den Murray Hill hinuntergeflossen, bevor Sidney Darlington 1953 die Idee mit dem gleichnamigen Transistor hatte. Auch wenn es viele Romane und Erzählungen gibt, die Technik und Elektronik auf eine spektakuläre Weise zum Thema haben, tendieren solche Elaborate eher dazu, die Unwissenheit der Schriftsteller zum Thema aufzuzeigen. Stellen Sie sich aber dann meine Überraschung vor, als ich vor kurzem einen Kriminalroman aus dem Polizeirevier 87 las, Fuzz von Ed McBain aus dem Jahre 1968. Ich sah mich plötzlich mit Schaltplänen konfrontiert und einer ausführlichen und plausiblen Erklärung, wie alles funktionieren sollte. Bilder in Romanen sind ohnehin selten, und wie hoch ist die Wahrscheinlichkeit, dass es sich auch noch um eine Elektronikschaltung handelt?

(160404-b)



Fuzz von Ed McBain (1968)





Der degradierte Schaltungs-Opamp LM0901A1411090451C.

Bemerkenswerte Bauteile

Der **Operationsverstärker LM0901A1411090451C** ist ein leistungsschwacher, degradierter Makro-Power-Schaltungsoperationsverstärker mit einer Leerlauf-Verlustleistung von weniger als 0,553 W bei V_s = ±1 pV und weniger als 200 W bei V_s = ±2 pV. Die offene Schleifenverstärkung ist größer als 0,001 k und der Eingangs-Biasstrom beträgt typischerweise 200 A. Lesen Sie mehr über dieses phantastische Bauteil unter:

© March 1878, Nominal Semidestructor Corporation

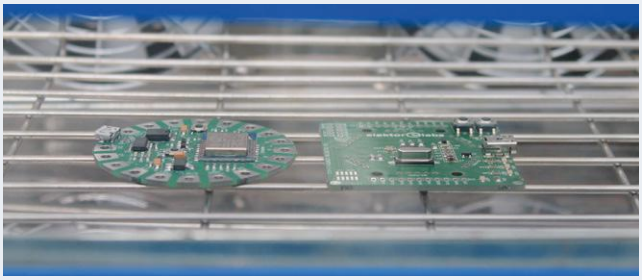
https://goo.gl/jFf75N

(160404-c)

Erweitern Sie Ihre Reflow-Fertigkeiten!

Sie haben es vielleicht nicht bemerkt, aber das Hauptthema dieser Ausgabe ist die Leiterplatte oder Platine. Da Leiterplatten und Löten eng zusammenhängen, ist die Kunst des Lötens eine wichtige Fähigkeit auch für den, der die Platine entwirft, da die Technik des Lötens Einfluss auf den Leiterplattenentwurf hat.

Wenn SMDs in einem Reflow-Ofen gelötet werden, muss die Temperaturentwicklung präzise verlaufen. Das thermische oder



Temperaturprofil muss stimmen, um gute Lötverbindungen zu gewährleisten. Temperaturprofile für die Bauteile finden Sie in den Datenblättern, aber nicht alle Komponenten benötigen das gleiche Profil. Dennoch müssen alle Teile, unabhängig von ihrer Größe und Masse, innerhalb eines definierten Zeitraums die Minimaltemperatur erreichen, bei der der Reflow-Prozess beginnt, und die maximale Temperatur, die sie verkraften können, nicht überschreiten. Es ist leicht zu sehen, dass dies ein komplizierter Prozess ist.

Es gibt seit 2001 eine Norm für die thermische Profilierung namens IPC-7530. Sie ist leider veraltet, die Revision A soll demnächst aufgelegt werden. Eigentlich ist dies kein Update, sondern eine komplett neue Norm. Das neue Dokument bietet eine Vielzahl von Informationen, von Definitionen bis zu Vergleichen von Lötmitteln, von Techniken zur Messung thermischer Profile bis zur Auswahl eines Reflow-Ofens und zum Verständnis von Reflow-Lötfehlern.

Der Zweck dieser Norm ist es, den Verantwortlichen, die thermische Profile entwickeln, nützliche und praktische Informationen zur Verfügung zu stellen, um akzeptable Zinn-Blei- und bleifreie Elektronik-Baugruppen zu entwickeln. Offiziell richtet sich die Norm an Führungskräfte, Entwicklungs- und Prozessingenieure sowie Techniker, die sich mit Massenlötprozessen beschäftigen. Aber auch Hobbyisten und andere Gelegenheits-Reflowlöter können dieses Dokument nutzen, um ihre Reflow-Fähigkeiten deutlich zu verbessern. Wer einen Ofen während der Abkühlphase öffnet, um das Board schneller abkühlen zu lassen, sollte noch schneller dieses Dokument lesen, um herauszufinden, warum er sich besser in Geduld üben sollte.

Als dieser Artikel geschrieben wurde, stand der Entwurf des Dokuments vom August 2016 kostenlos zum Download unter https://goo.gl/l88pjX zur Verfügung.

(160404-e)

Muss ich haben!

Wenn Sie jemals in einer Situation waren, in der es nur noch mit einem Zahnarzt-Mundspiegel möglich war, ein Gerät an einer unzugänglichen Stelle zu reparieren, könnte diese Brille genau das sein, was Sie brauchen. Mit ihren Prismen anstelle von normalen Linsen ist es möglich, dass man nach unten blickt, während man geradeaus sieht, oder Dinge in Löchern beobachtet, in die der Kopf nicht hineinpasst. Sie sind für das Lesen am Strand gedacht, damit man seinen dicken Kopf nicht heben muss, oder für das diskrete Ausspionieren anderer Leute. Wie wäre es, wenn man eines der Prismen um 180° drehen würde? Damit ließe sich gleichzeitig auf und ab schauen! Oder eine um 90° nach links und die andere nach rechts drehen und die interessante Erfahrung machen, wie man so als Esel in die Welt schaut? Erhältlich über das Internet, Stichwort Prismenbrille.

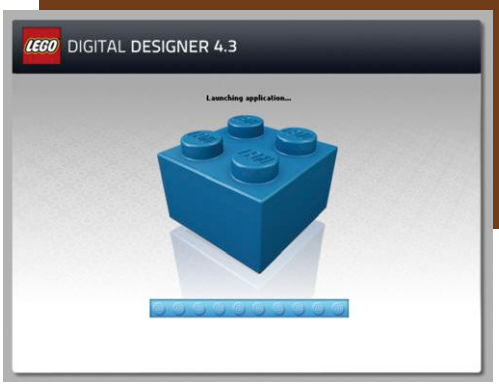
(160404-d)

Quelle: Jannis Hermanns, https://jann.is/

Just another brick...

Wenn Sie keinen Zugang zu CNC-Maschinen und Laserschneidemaschinen haben und auch nicht so fit im Bohren/Fräsen/Schneiden sind, können Sie jederzeit Legos verwenden, um ein Gehäuse für Ihre Schaltung zu bauen. Viele Leute tun dies zum Beispiel für ihren Arduino oder Raspberry Pi, und ich habe sogar Ätzwerkzeuge mit den kleinen bunten Steinchen gebaut. Jannis Hermanns aus Berlin hat einen kleinen Macintosh-Classic-Klon aus Legosteinen, einem Raspberry Pi und einem E-Paper-Display entworfen. Um sein gemauertes Gehäuse zu entwerfen, hat er die Hilfe des Lego-Digital-Designers (LDD) in Anspruch genommen. LDD kann kostenlos heruntergeladen werden von **http://ldd.lego.com**.

(160404-f)



Möchten Sie mitmachen? Bitte senden Sie Ihre Kommentare, Anwendungen, Tipps und Tricks an labs@elektor.com!

5 V aus erschöpften Alkali-Batterien

Den letzten Tropfen „Saft“ rausquetschen

Jeden Tag landen Tonnen von vermeintlich leeren 1,5-V-Alkali-Batterien (Primärzellen) im Müll. Es hat mich schon immer gestört, dass Geräte mit nur teilweise entladenen Zellen oft nicht mehr arbeiten. Also entwickelte ich eine Schaltung zur Spannungserhöhung solcher Zellen auf 5 V. Diese Spannung kann zur Versorgung einer LED, eines stromsparenden Mikrocontrollers oder einer IR-Fernbedienung eingesetzt werden, die nicht viel Energie benötigt. Bis dass das letzte Elektron uns scheide...

Von **Juan Canton** (Mexiko)

Dieser selbstschwingende Step-up-Konverter liefert an Spannungen bis hinab zu 0,8 V eine Ausgangsspannung von knapp 5 V. Der Wirkungsgrad von etwa 60 % bei einem Ausgangsstrom von 5 mA ist zwar nicht beeindruckend, aber für so niedrige Spannungen durchaus beachtlich. Wichtiger aber war mir, möglichst viel Energie aus alten 1,5-V-Zellen herauszupressen. Zudem war das Ganze ein interessantes Experiment.

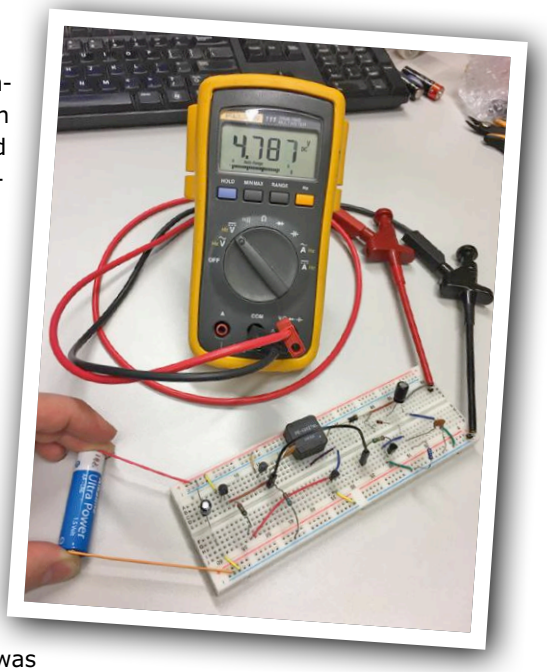
Die kritischen Punkte der Schaltung sind L1 und die Transistoren. Bei Letzteren könnte man annehmen, dass man die amerikanischen TUN und TUP der Typen 2N3906 und 2N3904 in Europa einfach durch BC560- und BC550-artige Exemplare ersetzen kann. Doch so einfach ist das nicht. Streukapazitäten und diverse andere Bauteile-Eigenschaften geben wohl den Spielverderber.

Im Prinzip wird der Oszillator um T1-T2-L1-C1 bis zu einem gewissen Grad von T4 gesteuert und so die Ausgangsspannung an K1 beeinflusst. Die Spule L1 speichert geschaltete Energiehäppchen, die dann in C2 landen, der zur Glättung beiträgt.

Interessant ist auch D2. Der Typ 1N4732A ist ein Parallelregler von NXP für 4,7 V bei I_{fwd} = 53 mA bzw. nur 10 µA (I_{rev}) bei V_{rev} = 1 V. L1 ist eine „richtige“ Spule und keine Drossel. Laut des Berichts vom Labor-Praktikanten funktionieren etwas höhere Werte sogar leicht besser. No experiment – no fun! Beim flachen und kompakten Typ PE52627NL von *Pulse Engineering* (#1209550 @ Farnell) wird ein Widerstand von 0,78 Ω angegeben. Für D1 kommt mit dem Typ 1N5817 eine 1-A-Schottky-Diode (aufgrund des niedrigen Spannungsabfalls) zum Einsatz. Eine 1N400x aus der Grabbelkiste bringt keine guten Resultate.

Praktischerweise steigt die Ausgangsspannung bei höherer Betriebsspannung nicht an, sodass die Last dann nicht ins Schwitzen kommt. Bei steigender Ausgangsspannung unterbindet nämlich die Regelung aus T2, T4 und T3 zunächst weitere Oszillationen. So bleibt der Ausgang (in Grenzen) stabil. ⏮

(150724)



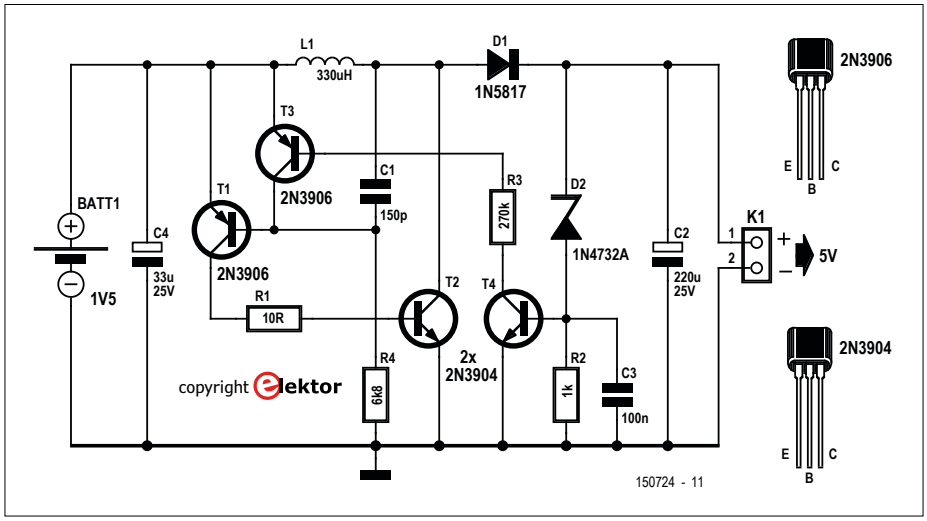
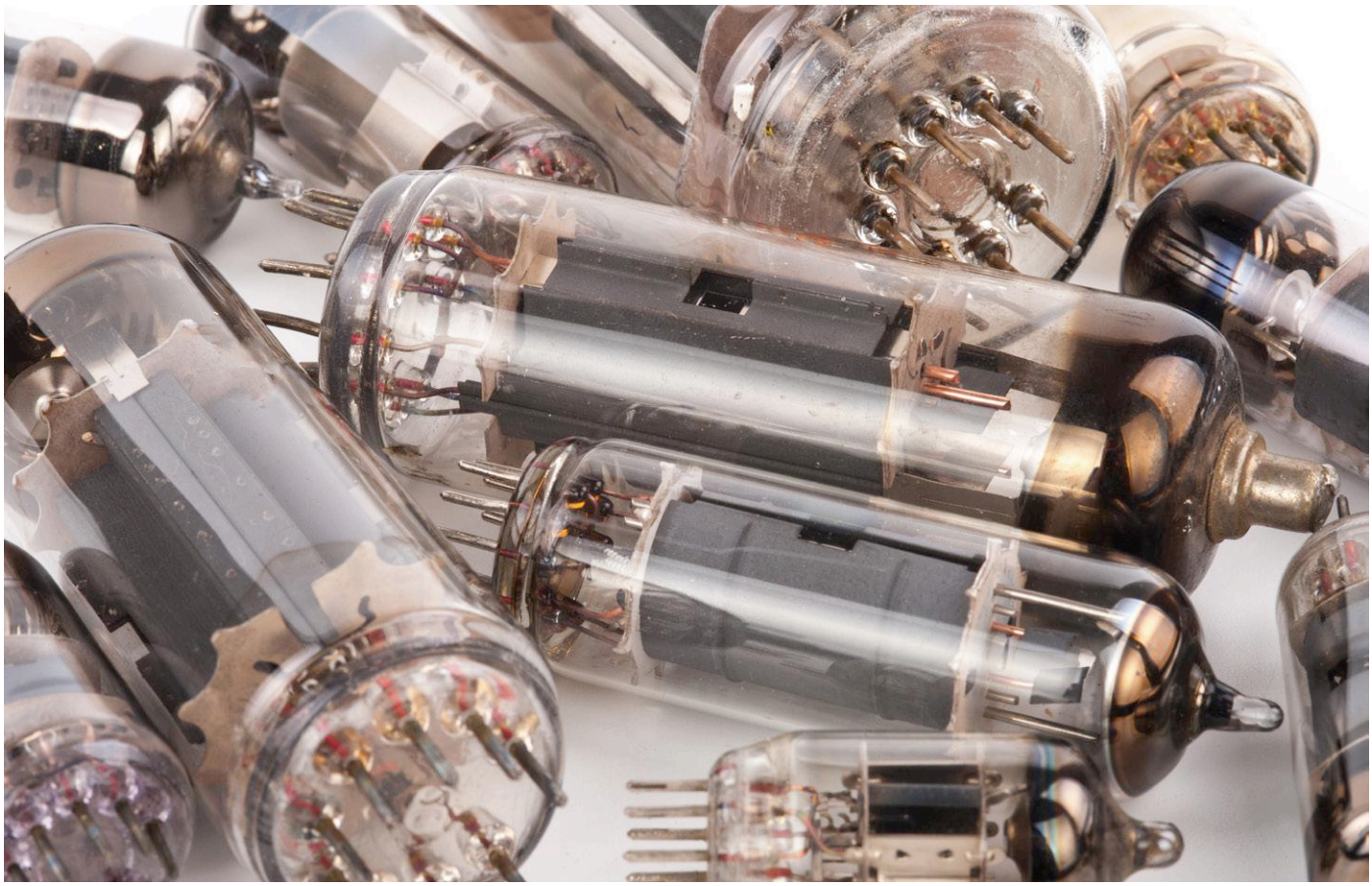


Bild 1. Ein experimenteller 5-V-Step-up-Konverter für teilentladene 1,5-V-Batterien.

Röhren-Kennlinienschreiber

Remake eines Tektronix 570



Um ausgeschlachtete Röhren testen zu können, benötigte der Autor ein Prüfgerät. Weil aber ein echter Tester unerreichbar war, entschied er sich zur Selbsthilfe – und entwarf einen Kennlinienschreiber auf Basis eines modernen Mikrocontrollers. Das Gerät kann den Kathodenstrom als Funktion der Anoden- oder der Gitterspannung darstellen.

Von **Charles van den Ouweland** (NL)

Sie können auch nichts wegschmeißen? Hoffnungslos defekte „retronische" Geräte, die Sie der Familie, Freunden oder Nachbarn abschwatzen, werden bis auf das kleinste vielleicht noch irgendwann brauchbare Teil ausgeschlachtet? Dann haben auch Sie sicherlich eine große Kiste mit staubbedeckten Röhren irgendwo im Keller stehen!

Was tun Sie mit all diesen aus Altgeräten geretteten Teilen? Natürlich verwenden - für schnelle und schmutzige Experimente sind sie oft noch ausgezeichnet brauchbar. Widerstände, Kondensatoren, Spulen und ähnliches können schnell durchgemessen werden und auch bei Transistoren und Dioden kann man leicht fest-

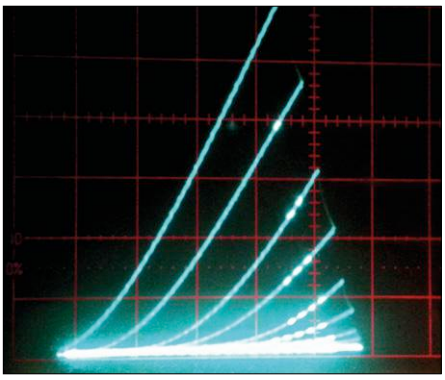


Bild 1. So sehen die vom Röhrentester produzierten Kurven aus.

Bild 2. Das originale Tektronix 570 ist ein wahrer Koloss von Messgerät.

stellen, ob sie noch funktionieren. Bei Vakuumröhren aber ist dies nicht so einfach. Ok, schwere Schäden sind mit dem bloßen Auge erkennbar und ein möglicher Kurzschluss zwischen den Elektroden kann eventuell mit einem Ohmmeter nachgewiesen werden, aber damit hört es auch schon auf. Es hat wenig Sinn, eine Schaltung mit einer Röhre zu bauen, wenn man nicht weiß, ob diese auch in Ordnung ist. Denn wenn die Schaltung nicht funktioniert, wer trägt daran die Schuld? Die Röhre? Die Schaltung? Beides vielleicht?

Der Kennlinienschreiber Tektronix 570

Selbst wenn Sie kein Projekt mit Röhren planen, können Sie immer noch versuchen, sie zu verkaufen. Bei einer Recherche auf eBay stieß der Autor auf einen französischen Anbieter, der jeder seiner Röhren im Angebot ein Foto ihrer Charakteristik zur Seite stellte, das von einem alten Kennlinienschreiber Tektronix 570 aufgenommen worden war.

Um diese Kennlinien zu ermitteln (**Bild 1**), legt der Tester verschiedene Gitterspannungen an die zu testende Röhre (0 V, -1 V, -2 V und so weiter) und ändert dann die Anodenspannung von Null bis zu einigen hundert Volt. Auf dem Bildschirm seiner Kathodenstrahlröhre zeigt der Tektronix 570 dann den Kathodenstrom bei verschiedenen Gitterspannungen als Funktion der Anodenspannung. Diese Kurvenschar zeigt, ob die Röhre für eine bestimmte Anwendung brauchbar ist.

Der erste Gedanke des Autors war „Das will ich auch haben!“. Das Problem ist nur, dass der Tektronix 570 (**Bild 2**) ein Sammlerobjekt worden ist. Wenn das Gerät irgendwo irgendwann erhältlich ist, müssen Sie dafür tief in die Tasche greifen. Außerdem ist der 570 ein wahrer Koloss mit Abmessungen von 42 x 33 x 62 cm und einem Gewicht von ungefähr 35 kg.

Wie bei jedem rechtschaffenen Elektroniker führte das unmittelbar zum zweiten Gedanken: „Das kann ich auch selber bauen!“. Doch auch wenn man im Netz [1] das komplette Handbuch mit Schaltplänen und Stücklisten findet, dürfte man bei der Bauteilbeschaffung schnell an seine Grenzen (beziehungsweise die des Elektronikmarktes) stoßen. Doch mit moderner Halbleiter- und Mikrocontroller-Technik und der Unterstützung eines normalen Oszilloskops sollte man ein Messgerät mit den Funktionen des Tektronix zu einem Bruchteil der Kos-

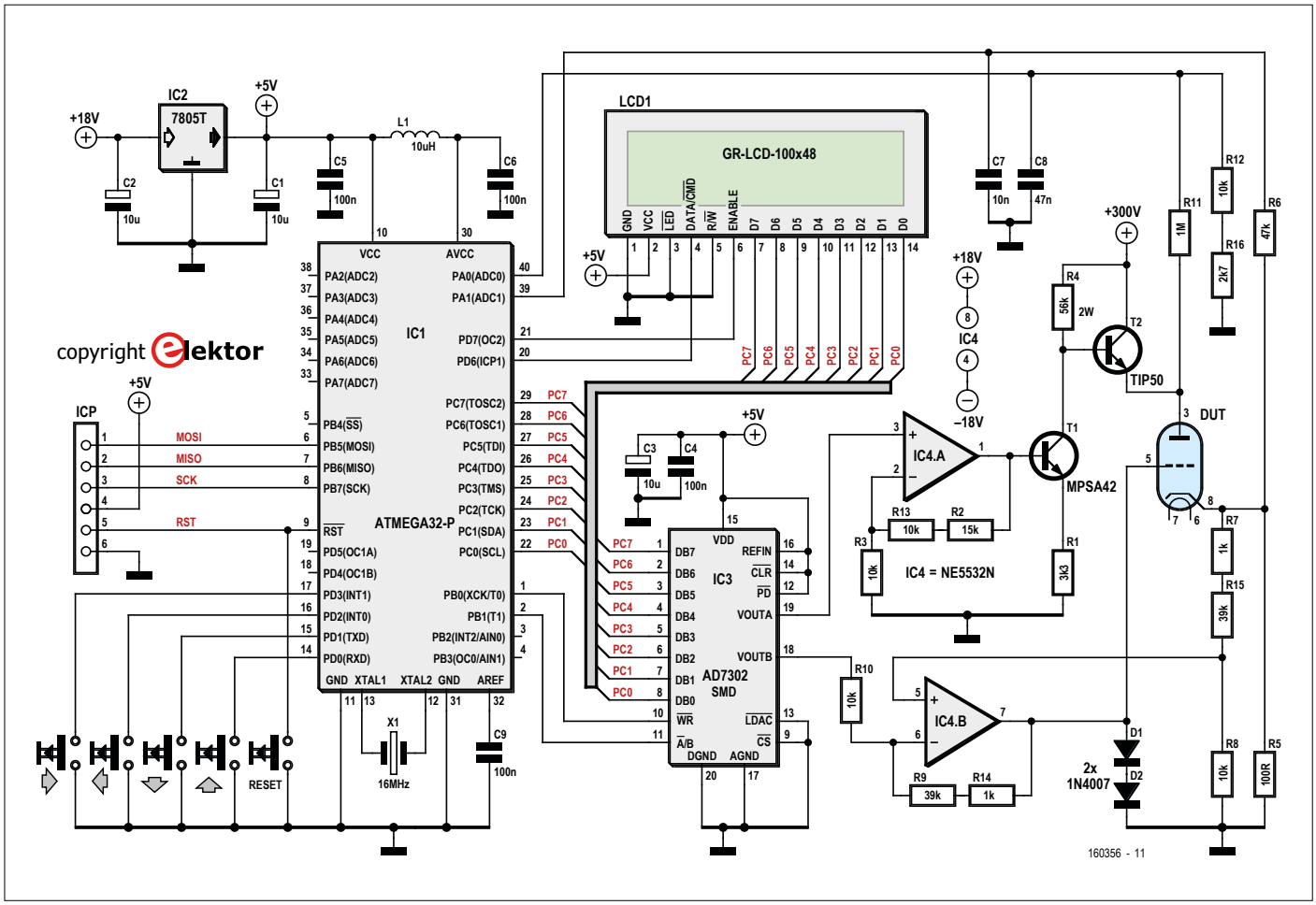


Bild 3. Die Schaltung des Röhrentesters.

ten (und des Gewichts) bauen können: Die Idee für das „Neptronix 570“ (so der spätere Name) war geboren!

Die Idee

Die Schaltung ermöglicht zwei Arten von Messungen:

- Variation der Gitterspannung bei verschiedenen festen Werten der Anodenspannung
- Variation der Anodenspannung bei verschiedenen festen Werten der Gitterspannung.

Dies alles wird problemlos von einem Mikrocontroller gesteuert. Als Benutzerschnittstelle dienen eine kleine LCD-Anzeige und vier Drucktasten. Die Software ist so ausgelegt, dass sie die gemessenen Charakteristiken auch auf dem Display anzeigt, anstatt auf den Bildschirm eines externen Oszilloskops zurückgreifen zu müssen. Das Neptronix 570 war und bleibt aber ein Versuchsaufbau auf Steckbrettern, eine richtige Platine ist nicht vorgesehen.

Die Schaltung im Überblick

Das Schaltbild des Kennlinienschreibers ist in **Bild 3** zu sehen. Der Mikrocontroller auf der linken Seite (IC1) ist ein ATmega32P. Dieser Typ wurde gewählt, weil er eine große Zahl von I/O-Leitungen bietet, sich in einem DIP-Gehäuse befindet, preiswert ist und – weil der Autor noch ein paar davon in der Schublade hatte.
Links vom Controller befinden sich die ICP-Programmierbuchse und die vier Bedientasten (plus eine Reset-Taste). Der 8 Bit breite Datenbus des Controllers ist sowohl mit dem LCD (LCD1) als auch mit einem Zweifach-D/A-Wandler (IC3) verbunden, der die verschiedenen Testspannungen für die Röhre liefert.
Die analogen Ausgangsspannungen des Wandlers gelangen über zwei Opamps zur Röhre; die Anodenspannung über den oberen Opamp (IC4.A) und zwei Hochspannungstransistoren, die Gitterspannung über IC4.B.

Spannungsversorgung

Bekanntlich arbeiten Röhren ja mit recht hohen Spannungen, wie man auch dem Schaltplan entnehmen kann. Zunächst wollen wir sehen, woher diese hohen Spannungen überhaupt kommen.
Die Versorgungsspannung für die digitale Elektronik und die Anzeige (5 V) wird durch einen simplen 7805T (IC2) aus einer Eingangsspannung von +18 V erzeugt, die aus einem Labornetzgerät stammt. Achten Sie darauf, dass der 7805 wegen der hohen Verlustleistung ausreichend gekühlt werden sollte! Das gleiche Labornetzgerät liefert auch eine negative Spannung von -18 V, um die Opamps symmetrisch zu versorgen. Da jeder Elektroniker wohl ein (Doppel-) Labornetzgerät besitzt und sich damit auskennt, brauchen wir nicht im Detail darauf einzugehen.
Die Sache mit der Hochspannung ist etwas komplizierter. Nicht jeder besitzt eine einstellbare Hochspannungs-Gleichstromversorgung (der Autor auf keinen Fall), so dass eine andere Lösung gefunden werden muss. Ein allzu großes Problem ist das zum Glück nicht, wie die Schaltung in **Bild 4** zeigt. Es ist eine Standardanwendung des Step-up-Wandlers MC34063AP.
Der Autor hat seiner Natur gemäß das Wrack eines Canon-Pixma-Druckers zerlegt und ein Netzteil vorgefunden, das für den Step-up-Wandler das nötige „Futter“ von 32 V zur Verfügung stellt. Man kann ein beliebiges Netzteil verwenden, aber wenn die verfügbare Eingangsspannung (viel) niedriger ist, kann der MC34063AP Schwierigkeiten bekommen, die gewünschte hohe Ausgangsspannung zu erzeugen. In diesem Fall hilft ein Spannungsverdoppler vor dem Eingang der Wandlerschaltung.

Einige Details

Der doppelte DA-Wandler IC2, ein AD7302, ist leider nicht zu einem akzeptablen Preis im DIP-Gehäuse erhältlich. Der Autor hat deshalb die SMD-Version dieses ICs auf ein Adapterplatinchen

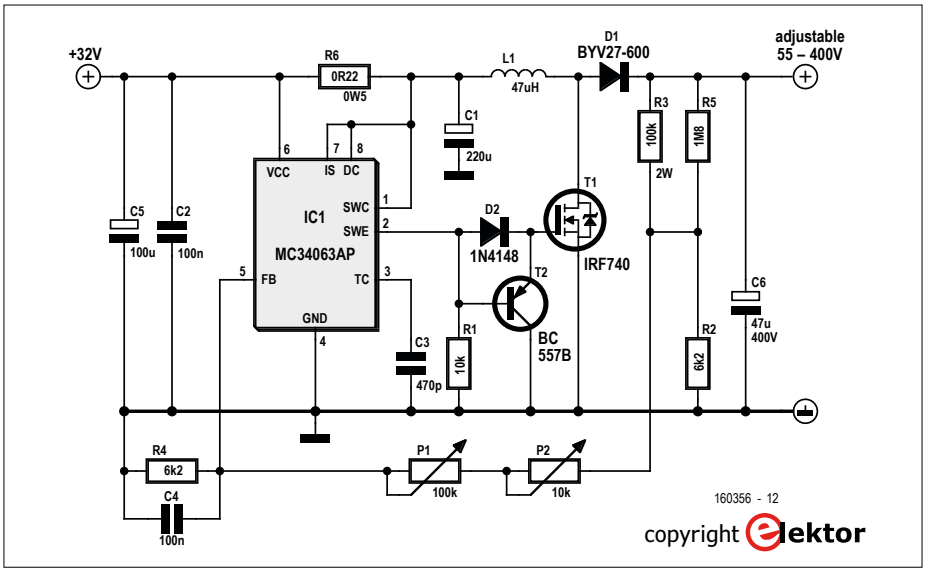


Bild 4. Die Eingangsspannung von 32 V für die Hochspannungsversorgung wird von einem alten Druckernetzteil geliefert. Außerdem ist noch eine Versorgung für die Heizung nötig.

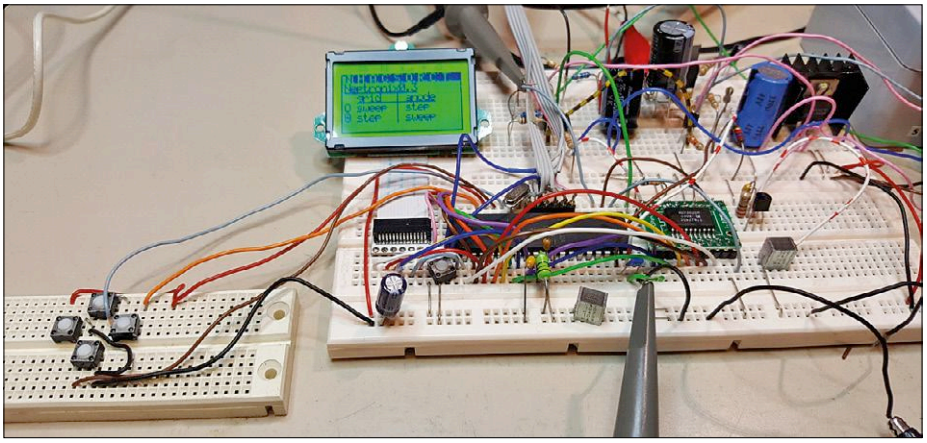


Bild 5. Der Röhrentester auf dem Steckbrett.

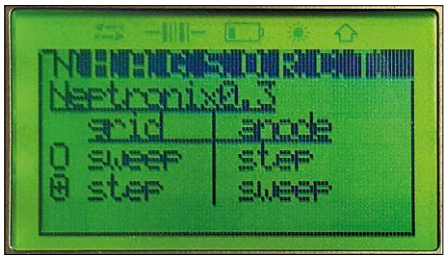


Bild 6. Der Start-Bildschirm.

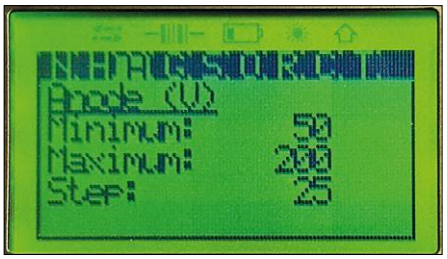


Bild 7. Der Anoden-Bildschirm.

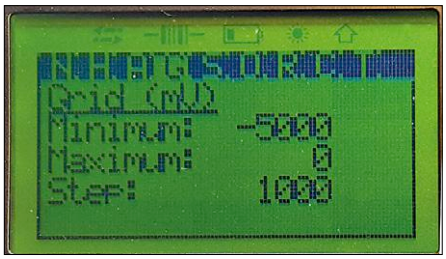


Bild 8. Der Gitter-Bildschirm.

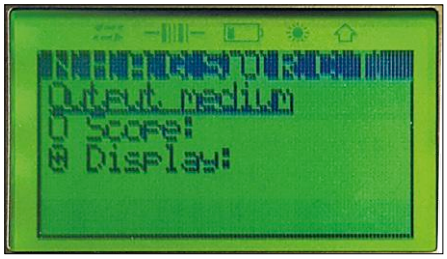


Bild 9. Der Output-Bildschirm.

Bild 10. Der Run-Bildschirm.

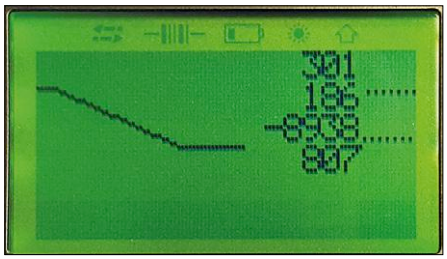


Bild 11. Anzeige der Kalibrierung.

Bild 12. Berechnung der Transkonduktanz.

montiert, damit es problemlos auf einem Steckbrett untergebracht werden kann. Die Anodenspannung für die „Röhre under test“ beträgt maximal etwa 320 V. Opamp IC4.A multipliziert deshalb die Spannung vom DA-Wandler (0...5 V) mit dem Faktor 3,5 und T1 nochmals mit dem Faktor 17. Der Transistor T2 sorgt dafür, dass ausreichend Leistung zur Verfügung steht.

Wenn wir nur die Anodenspannung variieren (sweepen) wollten, so wäre es nicht notwendig, die genaue Verstärkung von IC4.A und T1 zu kennen. Es gibt jedoch auch einen umgekehrten Modus, nämlich das Variieren der Gitterspannung bei fester Anodenspannung. Deshalb muss auch die Anodenspannung präzise gesteuert werden können. Aus diesem Grunde sorgt der Spannungsteiler aus R11, R12 und R16 dafür, dass ein Achtzigstel der Anodenspannung zum AD-Eingang des Mikrocontrollers (ADC0) zurückgeführt wird. Der Controller misst und vergleicht und kann so die Anodenspannung kalibrieren.

Der andere Ausgang des DA-Wandlers ist an Opamp IC4.B angeschlossen. Da die Gitterspannung negativ sein muss, wandelt dieser Opamp die 0...5 V an VOUTB des DAC in einen Bereich von 0...-18 V. Die Dioden D1 und D2 schützen die Schaltung vor zu hohen Spannungen. Um den Kathodenstrom zu messen, wurde der Widerstand R5 (100 Ω) zwischen Kathode und Masse geschaltet. Die Spannung über R5 gelangt über R6 zum zweiten AD-Eingang (ADC1) des Mikrocontrollers. Um den Spannungsabfall über R5 zu kompensieren, wird ein Fünftel der Kathodenspannung über den Spannungsteiler R7/R15/R8 zum nicht-invertierenden Eingang des IC4.B zurückgeführt.

Bild 5 zeigt den Röhren-Tracer/Tester während der Entwicklungsphase.

Benutzerinterface

Das Benutzerinterface des Röhrentesters besteht aus einem LC-Display und vier Drucktasten (*up, down, left, right*). Das Display zeigt eine Reihe von Bildschirmen, in denen verschiedene Einstellungen vorgenommen werden können. Mit der *Left*- und der *Right*-Taste kann man zwischen den verschiedenen Bildschirmen navigieren, mit der *Down*-Taste wird der entsprechende Bildschirm geöffnet. Dann lassen sich mit *left* und *right* Änderungen der Einstellungen auf diesem Bildschirm vornehmen. Hier eine kurze Übersicht der einzelnen Bildschirme:

Start (N)

Im Eingangsbildschirm (**Bild 6**) wird mit *up* und *down* zwischen den beiden Betriebsarten umgeschaltet:
Variation der Gitterspannung für verschiedene Stufen der Anodenspannung (*step plate/anode, sweep grid*)) oder
Variation der Anodenspannung für verschiedene Stufen der Gitterspannung (*step grid, sweep plate/anode*).
Mit *right* kann man zum nächsten Bildschirm gelangen.

Anode (A)

Auf diesem Bildschirm (**Bild 7**) werden die minimale und maximale Anodenspannung sowie die Schrittweite für die Anodenspannung eingestellt. Mit *up/down* kann navigiert, mit *left/right* der Wert geändert werden. Die Spannungsstufe wird benötigt, wenn im Startbildschirm der Modus *step plate/anode, sweep grid* eingestellt ist. Dann bestimmt der eingestellte Wert, um wieviel die Minimal- und Maximalspannung erhöht respektive vermindert wird.

Grid (G)

Hier werden die minimale und maximale Gitterspannung sowie die Gitter-Spannungsstufe eingestellt (**Bild 8**). Diese Gitter-Spannungsstufe kommt im Modus *step grid, sweep plate/anode* zum Tragen und bestimmt, inwieweit Minimum und Maximum erhöht respektive verringert werden.

Output (O)
Hier können für die Ausgabe ein Oszilloskop oder das LC-Display ausgewählt werden (**Bild 9**).

Run (R)
Drücken Sie die *down*-Taste (**Bild 10**), um den Messvorgang zu starten. Wenn in *Output* das LC-Display ausgewählt ist, werden die Ergebnisse dort angezeigt. Auf der horizontalen Achse werden die Sweep-Parameter angezeigt.
Wenn ein Oszilloskop zur Anzeige der Charakteristiken gewählt ist, muss es wie folgt angeschlossen werden. Im Modus *sweep grid, step anode*: x-Achse am Gitter, y-Achse an der Kathode; im Modus *sweep anode, step grid* liegt die x-Achse an der Anode und die y-Achse an der Kathode.

Im LCD-Modus werden die Messungen relativ langsam ausgeführt, weil die Firmware den Mittelwert einer Anzahl von Messungen berechnet, bevor sie das Ergebnis an den Bildschirm sendet. Im Modus Oszilloskop jedoch finden die Messungen recht schnell statt, um ein unruhiges, flackerndes Bild zu vermeiden.

Calibrate (C)
Nach einem Druck auf die *Down*-Taste erscheint dieses Fenster (**Bild 11**) mit einem Diagramm der Anodenspannung (0...320 V) in Abhängigkeit vom DAC-Wert (0 ... 255). Die numerischen Werte in dem Beispiel von Bild 11 bedeuten (von oben nach unten): die maximale Spannung beim DAC-Wert 0, die minimale Spannung beim DAC-Wert 255, Steilheit und der Schnittpunkt der Geraden mit der y-Achse.
Drückt man auf *up*, so verschwindet diese Grafik und der theoretische Kurvenverlauf (mit denselben Achsen) erscheint.

Transconductance (T)
Dieser Bildschirm (**Bild 12**) zeigt die Steilheit S oder Transkonduktanz. Zunächst wird die Anodenspannung im *Anode*-Bildschirm auf ihren Maximalwert eingestellt, ebenso die Gitterspannung im *Grid*-Bildschirm. Dann werden der Strom I1 gemessen, anschließend die Gitterspannung um eine Stufe reduziert und der Strom I2 gemessen. Die Steilheit berechnet sich nach:

$$S = (I_1 - I_2) \text{ / grid step voltage} \qquad [A / V]$$

In dem Beispiel aus Bild 12 betragen die Anodenspannung 200 V und die Gitterspannungen 0 V und -1 V. Die Messwerte sind I_1 = 4,3 µA und I_2 = 2,7 µA. Daraus resultiert eine Steilheit von 1,6 mA/V.

In einer späteren Firmware-Version sollen noch die Bildschirme H (Heizspannung/Heizstrom) und S (Schirmgitter) hinzukommen. Die Firmware des Röhrentesters, die mit einem Standard-AVR-Programmer über die Programmierschnittstelle ICP in den Mikrocontroller geladen wird, kann in der aktuellen Version von der Elektor-Labs-Seite [2] heruntergeladen werden. An dieser Stelle werden Sie stets über neue Entwicklungen und Updates informiert und Sie können auch ihre eigenen Anmerkungen und Ideen beisteuern. ⏮

(160356)

Über den Autor

Charles van den Ouweland (Breda, 1963) studierte an der Technischen Universität in Eindhoven Elektrotechnik. Er arbeitet zurzeit als Software-Architekt beim niederländischen Schienennetzbetreiber ProRail, beschäftigt sich aber in seiner Freizeit viel mit Elektronik – vor allem mit der Kombination von modernen Mikrocontrollern mit „altertümlichen“ Röhren.

Weblinks

[1] www.jogis-roehrenbude.de/Roehren-Geschichtliches/Roe-Pruefer/Curve-Tracer/TEC-570/TEC-570.htm

[2] www.elektormagazine.com/labs/vacuum-tube-curve-tracer-remake-of-tektronix-570

Anzeige

HAMMOND MANUFACTURING®

Hand Held Gehäuse
Standard und Wasserdicht

www.hammondmfg.com/1553.htm
www.hammondmfg.com/1553W.htm

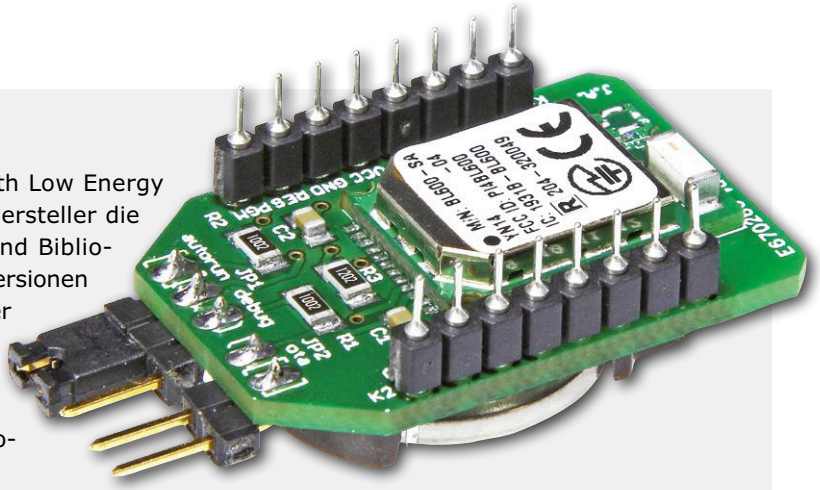
01256 812812
sales@hammondmfg.eu

Projekt 2.0

Korrekturen, Updates und Leserbriefe

e-BoB BL600 (1)

Elektor 03/2015, S. 34 (140270)



Seit der erste Artikel über das Laird BL600 Bluetooth Low Energy Modul im März 2015 veröffentlicht wurde, hat der Hersteller die Software für dieses Modul aktualisiert (Firmware und Bibliotheken). Unglücklicherweise scheinen die Vorgängerversionen nicht länger verfügbar zu sein. Das Beispiel, das in der Artikelserie von 2015 präsentiert wurde, basiert auf der 1.5.7.0 Revision 5 Firmware und den dazugehörigen Bibliotheken (libraries). Es könnte sein, dass es nicht mit der neuen Firmware und den neuen Bibliotheken kompatibel ist.
Auf jeden Fall ist es wichtig, nur die Bibliothek zu verwenden, die zur jeweiligen BL600 Firmware/Revision passt. Falls Sie einen Fehler „BL600: Cross Compiler not found“ erhalten, sind ihre Modul-Firmware und die heruntergeladene Bibliothek nicht kompatibel.
Die AT-Kommandos zur Identifizierung der Firmware und Revision des BL600s (verbinden Sie das Modul zuerst via serieller Schnittstelle mit dem PC) sind:

- firmware: AT I 3<Enter>
- revision: AT I 0<Enter>

Siehe auch den Artikel „eBoB BL600 (2)“ (www.elektormagazine.de/150014) für weitere Details zur Firmware.
Der Artikel „Bluetooth-2/LE-Bridge“ (www.elektormagazine.de/150753) enthält die Vorgehensweise zur Programmierung der BL600-Firmware. Die Reprogrammierung des BL600 erfordert einen Segger J-Link EDU (oder einen kompatiblen) Programmer.

Schwebende Lampen

Elektor 01-02/2017, S. 48 (160232)

Im Schaltplan auf Seite 51 sind bei IC3 (L293D) zwei Pinnummern vertauscht. Richtig ist: Der mit +V11 verbundene Pin trägt die Nummer 8 und der mit +5 V verbundene Pin hat die Nummer 16.
Siehe Datenblatt www.ti.com/lit/ds/symlink/l293.pdf.

Willkommen bei Share

Elektor 01-02/2017, S. 118 (160252)

Zur Frage von Elektor-Redakteur Thijs Beckers, wieso die Polarität des Netzsteckers bei Heizkesseln eine Rolle spielen kann, haben uns sehr viele Zuschriften erreicht, hierfür vielen Dank! Hier eine Auswahl der Leserbriefe:

Das Problem entsteht, wenn die symmetrische Spannungsversorgung vom Netz nicht symmetrisch an die nachfolgenden Schaltungen weitergegeben wird.
Ein Beispiel, das ich schon seit längerem kenne, sind Leuchtstoffröhren. Da der Start-Trafo als Spartrafo ausgelegt ist, spielt es eine Rolle, wie der Netzanschluss eingesteckt wird, d.h. ob die L- oder N-Seite an der Basis des Trafos liegt. In einer Stellung wird man im Dunkeln ein leichtes Glimmen der Röhre feststellen, das in der anderen Stellung verschwindet.

Lothar Freißmann

Das Problem wird vermutlich durch Elektrolyse verursacht. Es gibt immer etwas Gleichspannung (DC) auf der Netzleitung (AC), und das verursacht manchmal, dass sich auf Elektroden ein Belag bildet. Dies kann zu einem gewissen Grade durch Erdung des Gerätes behoben werden.
Wenn man den Stecker umdreht, wird dieser elektrolytische Effekt umgekehrt; allerdings wird es dann zu Ablagerungen auf der anderen Elektrode in der Therme kommen.
Ich habe hier in Kanada ein ähnliches Problem, nämlich eine Ablagerung von Kalzium in meinem Durchlauferhitzer. Wenn man ein paar Stunden Essig durchlaufen lässt, kann man damit den Kalk entfernen. Ich habe eine Bypassleitung zum Durchlauferhitzer gebaut, um diesen einfacher reinigen zu können: Mit einer Pumpe wird dann Essig rundgepumpt, so lange ich möchte.

Clinton Millet, Kanada

Im Handbuch meines früheren Heizkessels „Remeha Calenta“ war ein Parameter 43 „Phasenerkennung Netz“ beschrieben. Nach der Aktivierung soll bei falsch gestecktem Netzstecker ein Fehlercode angezeigt werden. Das habe ich natürlich ausprobiert und den Stecker mal andersherum eingesteckt. Das ist in Deutschland und Holland möglich, aber z.B. nicht Belgien und Frankreich, wo die Geometrie der Steckdosen es nicht erlaubt. Nach dem Umdrehen des Steckers wurde Fehlercode SU9 angezeigt: Stecker verkehrt herum gesteckt. In der Werkseinstellung des Kessels stand der Parameter 43 auf 0 = aus. Der Kessel hätte den falsch gesteckten Stecker also nicht erkannt.
Die Zündelektrode wird nach dem Zünden dazu verwendet, um mittels des Ionisationsstroms zu messen, ob der Kessel gut brennt. Im Handbuch steht, dass der Ionisationsstrom unter 3 µA betragen sollte. Die Elektrode ist daher präventiv in verschiedenen Service-Sets für den Kessel enthalten. Die Hochspannung wird mittels eines Schaltreglers erzeugt, der eventuell zusätzliche Störungen erzeugen könnte, wenn der Stecker falsch gesteckt ist. Das ist eventuell ein bisschen zu viel für einen so niedrigen Messstrom an einer fast abgenutzten Elektrode.

Peter van de Meerendonk

Anschlüsse eines Opamps...

Elektor 01-02/2017, S. 124 (160257)

Da ich in der IC-Entwicklung gearbeitet habe, kann ich mir folgende einfache Erklärung für die unterschiedliche Belegung eines solchen ICs vorstellen, das sicher wie „Schüttgut“ hergestellt wird: Es gibt wichtige Abnehmer, die die alternative Pinbelegung für ein eleganteres Design benötigen; der Hersteller muss dazu nur das Bonding und die Beschriftung mit geringem Aufwand umstellen.

Lothar Freißmann

MEMS-Mikrofon - für den Fledermaus-DetektorPLUS

Elektor 12/2016, S.75 (160083)

Im Artikel steht, dass spezielle Mikrofone für Ultraschall schwer zu bekommen sind.
Stimmt nicht :).
Als alter „Nacht-Stricker“ (wann denn sonst hat man Ruhe?) bin ich auch Fledermaus-Fan. Ich habe bereits mehrere US-Umsetzer aufgebaut: Zunächst rein digital (mit einem Teiler 7493), später auch linear heruntergemischt (mit SO42P). Hierfür benötigte ich natürlich auch vernünftig arbeitende Mikros.
Und dafür bieten sich diese, sehr günstig zu bekommenden Teile an:
www.ebay.de/itm/Ultrasonic-Wave-Module-HC-SR04-Ultraschall-Distanz-Sensor-fur-Arduino-51-AVR-PIC-/351313562395
Die Platine enthält nicht nur ein RX-Mikro, sondern auch einen Chip, den man auch für andere Experimente verwenden kann, sowie auch noch einen als „Lautsprecher“ optimierten US-Geber (alte Ultraschall-TV-Fernsteuerungen lassen grüßen!). Diese Platinen gibt es auf eBay auch noch günstiger als 5er Pack; und man muss die Teile darauf nicht mit der Lupe suchen. Es handelt sich um Electret-Einheiten, so dass man nichts großartig ändern muss, um sie mit aller Art Fledermaus-Detektoren betreiben zu können.
Und nun kommt noch ein Vorteil: Da die Öffnung des Mikros recht groß ist, eignet sich das RX-Teil auch für den Einsatz zusammen mit einem (aus steifem Papier selbst leicht zu bauenden) Parabolspiegel, so dass man sich eine Peilantenne bauen kann, um die Flattermänner und -frauen sehr genau orten zu können.
Um der Frage vorzubeugen: Ja, die Mikros funktionieren bis über 100 kHz, was allerdings meist nicht nötig ist, weil die meisten heimischen Fledermäuse nicht „so hoch arbeiten“.

Peter Krengel

MIC
K3
R20
C35
L15
R22
C38
L14
R21
C37
C34
R19
L12
C31
C40
L31
C39
L32
L34
L35
C13
R15
R14
R1
R2
C2
C1
R3
R18
47-16L
F7

FPGA/DSP-Board für Schmalband-SDR

Ein voll-programmierbares Empfänger/Sender-Basisboard

Von **Daniel Uppström** (SM6VFZ, Schweden) und **Ton Giesberts** (Elektor-Labor)

Dieses DSP-Board auf FPGA-Basis enthält alles, um die Basisbandverarbeitung von SSB/CW- und AM-Funksignalen durchführen zu können. Es löst viele der Probleme und Einschränkungen einfacher SDR-Plattformen und bietet eine perfekte Grundlage für einen Transceiver von der Kurz- bis zur Mikrowelle. Unser Board ist ideal für wissenschaftliche, Amateur- und andere ernsthafte Funkapplikationen.

Von der guten alten Zeit ...

Es war einmal eine Zeit, in der die Funktechnologie der Schwerpunkt fast aller elektronischen Hobbyisten und Ingenieure war. Wer seine eigene Ausrüstung baute und eine Funklizenz erwarb, war in der Regel auch an Elektronik interessiert. Mit dem technischen Fortschritt konnten oder wollten die meisten Amateurfunker aber nicht mithalten und kauften deshalb ihre Ausrüstung fix und fertig. Die hochmodernen Transceiver der 80er und 90er Jahre nutzten zudem viele komplexe analoge Baugruppen, so dass es für Bastler aufgrund der Kosten, des Aufwands und der schwierigen Bauteil-Beschaffung fast unmöglich wurde, etwas Ähnliches zu bauen.

Mit der Entwicklung des Internets entstanden neue Kommunikationswege. Die Vielfalt kommerziell ausgestatteter Funk-Transceiver für Amateure schrumpfte. Gleichzeitig wurden viele Kompromisse akzeptiert, um Funkgeräte, die möglichst viele Frequenzbänder abdeckten, auch noch kleiner und billiger zu machen. Darunter litten wesentliche Eigenschaften wie die Empfangsselektivität und saubere Signalübertragung. Teilweise wurde die mangelnde Qualität durch ausgefallene digitale Funktionen, Farbdisplays und so weiter kompensiert. Doch wer sich nicht vom hippen Aussehen und lustigen Gadgets beeindrucken ließ und lieber ein High-Performance-Funkgerät bevorzugte, begann, sich wieder mit dem Selbstbau zu beschäftigen.

INFOS ZUM PROJEKT

FPGA, DSP, SDR, Radio, Transmitter, Receiver, Amateurfunk

Einsteiger
Fortgeschrittener
➡ Experte

etwa 8 Stunden

SMD-Lötmöglichkeiten, JTAG-Programmer (USB Blaster), Quartus Lite

etwa 200 €

... zum Software Defined Radio

Doch vielleicht ist die Rückkehr zur (modernen) analogen Technik nicht die beste Lösung. Das Gebiet des Software Defined Radio (SDR), wo ein großer Teil der Funksignalverarbeitung digital per Software ausgeführt wird, kann interessanter sein. Dank der Leistungsfähigkeit moderner Computer und guter Analog-Digital-Wandler (ADC) ist die SDR-Technologie durchaus eine sinnvolle Alternative.

Viele Hobby- und Amateurfunkanwendungen verarbeiten digitalisierte Signale komplett in der Software auf einem PC. Trotz seiner Flexibilität ist dieser Ansatz eher ineffizient, da die Signalverarbeitung in Bezug auf Prozessoroperationen immer noch relativ teuer ist, wenn sie ohne Hardware-Multiplizierer oder ähnliche Peripherie auskommen soll. Viele bekannte SDR-Plattformen besitzen auch nur eine schlechte (oder gar keine) analoge Filterung vor der Digitalisierung und nur niedrig auflösende ADCs, was zu einer schlechten Leistung insbesondere bei starken Störsignalen führt.

Besser wäre es, ein gutes analoges Radio-Frontend zu bauen und die Signalverarbeitung einer programmierbaren Logik zu überlassen. Heutige *Field Programmable Gate Arrays* (FPGA) sind preiswert und enthalten eine beeindruckende Anzahl von Gattern, Speicherbits und hartcodierten Multiplikatorblöcken. Sie sind deshalb hervorragend für komplexe Signalverarbeitungsaufgaben geeignet.

Lasst uns eines bauen!

Dieses Projekt begann im Jahr 2013, als der Autor den ersten Prototyp baute und sich damit beschäftigte, VHDL-Code für DSP-Blöcke zu schreiben. Im Vergleich zu einem DSP ist es einfacher, eine komplexe Signalverarbeitungskette in programmierbarer Logik zu implementieren, da sich alle Schritte unabhängig voneinander gestalten lassen. Am Ende muss man nur noch die Signale zwischen den einzelnen Blöcken verbinden. Die Beschreibung dieses Entwicklungsprozesses geht zwar über den Rahmen dieses Artikels hinaus, aber der gesamte Code und alle anderen notwendigen Informationen stehen im Downloadpaket auf der Elektor-Projektseite frei zur Verfügung [1].

Allerdings ist die nützliche und leistungsstarke Kombination programmierbarer Logik und AD-Wandler noch kein vollständiges Funkgerät. Die FPGA/DSP- muss um eine Funkplatine ergänzt werden, die die Frequenzkonversion, die analoge Filterung und die Verstärkung durchführt. Ein solches Board wird in einem zukünftigen Artikel vorgestellt, eine Steuerplatine für die Abstimmung, die Lautstärkeeinstellung und weiteren Funktionen wird das Thema eines weiteren Artikels. Mit diesen drei Boards lässt sich ein komplettes Funkgerät aufbauen, das nur noch um eine Frontplatte und ein geeignetes Gehäuse ergänzt werden muss.

Das System kann auch von einem PC über einen Seriell-zu-USB-Adapter oder von einem Raspberry Pi, der über seine I²C- oder serielle Schnittstelle kommuniziert, gesteuert werden. Eine grafische Benutzeroberfläche, die in Python geschrieben wurde, demonstriert die Funktionalität.

Schaltungsbeschreibung

Auf der FPGA/DSP-Platine befindet sich ein 24-Bit-ADC für die Abtastung der Zwischenfrequenz im Empfangsmodus, ein FPGA des Typs Cyclone IV für die Signalverarbeitung, ein Hochgeschwindigkeits-DAC für den lokalen Oszillator und die Erzeugung des Sendesignals, eine Audio-Schnittstelle für Mikrofon und Lautsprecher, ein hochstabiler, hochpräziser Oszillator und eine I²C- oder UART-Schnittstelle für einen Host-Controller (**Bild 1**).

Das Herz (oder besser das Gehirn) der FPGA/DSP-Platine ist das FPGA IC4, ein

EP4CE10 aus der Cyclone-IV-Reihe von Intel (früher Altera). Dieser Chip kann für nahezu jede digitale Funktion konfiguriert werden. Bei der Inbetriebnahme sorgt die Firmware dafür, dass alle Gatter so konfiguriert werden, wie es ein externer Konfigurationsspeicher (IC5) vorgibt. Für den Audioein- und -ausgang gibt es einen Codec (Coder-Decoder) mit der Bezeichnung TLV320AIC20K (IC3). Er besteht aus zwei ADC/DAC-Kanälen mit 16 Bit Auflösung und einer maximalen Abtastrate von 25.000 Samples pro Sekunde (SPS). In dem Codec stecken auch ein digitales 8-kHz-Tiefpassfilter, ein Mikrofonverstärker und ein kleines Verstärkerchen, das an einem 8-Ω-Lautsprecher bis zu 250 mW liefern kann. Die Verstärkung/Dämpfung aller Audio-Eingänge und Ausgänge ist programmierbar. Der Lautsprecher wird an K2 angeschlossen, das Mikrofon an K3. Alle anderen Kanäle von IC3 sind an K4 angeschlossen und stehen beliebigen Zusatzgeräten zur Verfügung.

Der Haupteingang des Boards ist K1. Er erwartet ein differenzielles Signal, typischerweise im Bereich einiger hundert Kilohertz. Das Signal durchläuft einen Differenzverstärker (IC1) und ein diskretes Tiefpassfilter, um schließlich vom 24-Bit-ADC AD7760 (IC2) digitalisiert zu werden. Dieser Chip besitzt viele Anschlüsse zur Stromversorgung, sauber getrennt durch die verschiedenen Versorgungsspannungen und/oder passive Filter. Der ADC arbeitet mit einem Taktsignal, das vom FPGA zur Verfügung gestellt und vom Pegelwandler IC8 auf 5 V angehoben wird.

Es gibt auch einen schnellen Digital-Analog-Wandler (DAC) mit zwei Kanälen auf der Platine, einen DAC5672 (IC6). Seine beiden Differentialausgänge A und B werden von den Übertragern TR1 und TR2 in asymmetrische Ausgänge umgewandelt. Die Tiefpassfilterung erlaubt die Passage der DAC-Signale bis zu 50 MHz zu den SMA-Koaxialsteckverbindern K8 und K9. Für den Haupttakt auf der Platine ist der temperaturkompensierte 20-MHz-Quarzoszillator (TCXO) IC7 verantwortlich. Das Taktsignal wird dem FPGA zugeführt, das es dann an die peripheren ICs verteilt. Die Frequenz kann mit dem Poti P1 feinabgestimmt werden. An Verbinder K5 kann auch, falls einmal vorhanden, eine externe Taktreferenz angeschlossen werden. Dann kann TCXO durch T1 abgeschaltet werden. Die Signale vom TCXO und von der externen Referenz werden über ungepufferte Inverter (IC9, IC10), die als analoge Verstärker geschaltet sind, zum FPGA geführt und können daher leichte Spannungsschwankungen aufweisen.

Der Takt wird im FPGA geteilt, um ein niederfrequentes Signal für die LED1 zu erzeugen. Die LED zeigt an, dass der Oszillator „läuft“ und das FPGA konfiguriert ist. LED2 ist reserviert für zukünftige Aufgaben und hat jetzt noch keine Funktion.

Die 5-V-Versorgungsspannung des Boards wird an K10 angeschlossen und gelangt über Verpolschutz T2 zu den vier verschiedenen Low-Dropout-Spannungsreglern (LDO) IC11...IC14, die für verschiedene Komponenten auf der Platine die Versorgungsspannungen erzeugen: 1,2 V, 1,8 V, 2,5 V und 3,3 V.

Die Schnittstelle zum Hostcontroller besteht aus zwei unscheinbaren Stiften auf K7, die für die I²C- oder UART-Kommunikation vorgesehen sind. Diese Wahl

Eigenschaften

- FPGA + DSP + Audio CODEC
- Schmalband-SDR-Plattform
- Einsetzbar im Funkbetrieb auf nahezu allen Frequenzen zwischen 0 und vielen GHz
- Nach dem Überlagerungsprinzip
- Niedrige zweite ZF höher als 0 Hz
- Kompatibel zu Receive/Transmit-Datenverkehr (RX/TX)
- Weaver-SSB-Modulator/Demodulator

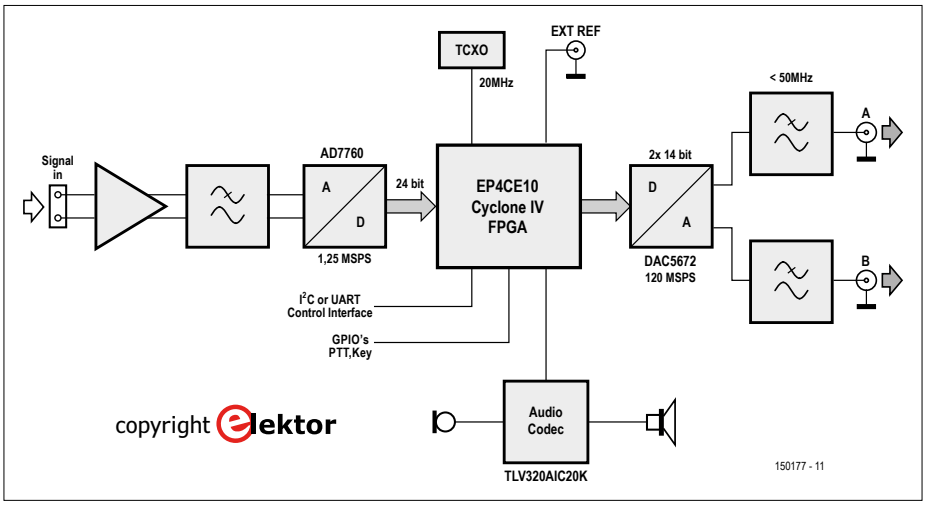


Bild 1. Blockschaltbild des FPGA-DSP-Boards.

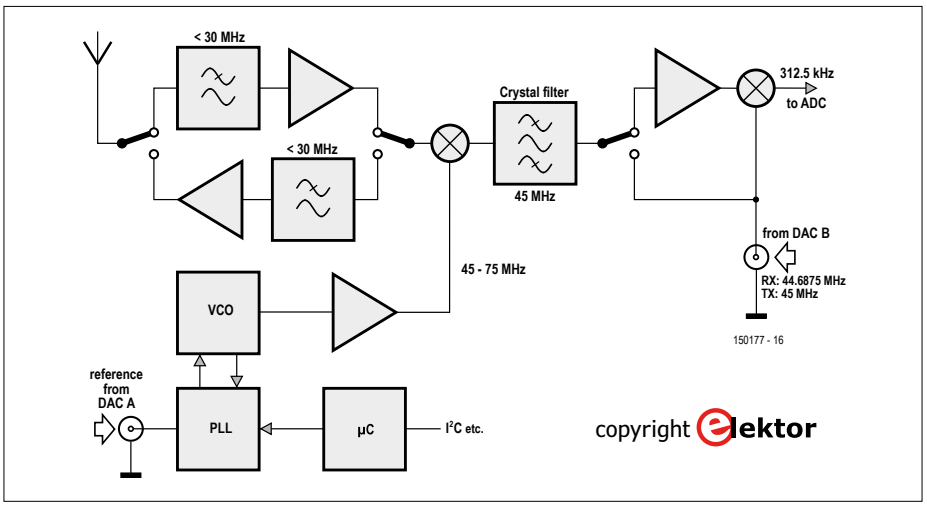


Bild 3. Blockschaltbild des Funk-Zusatzboards.

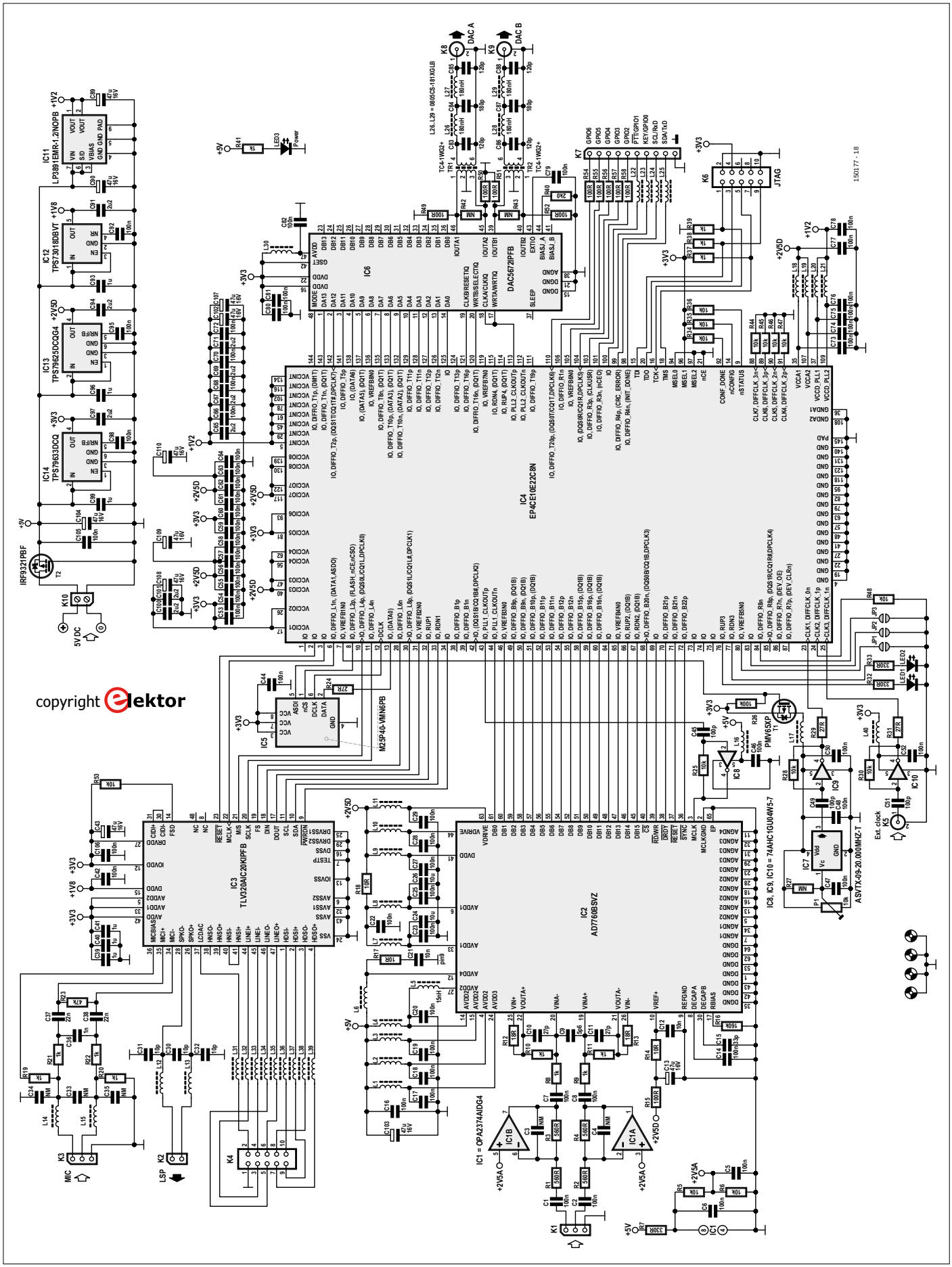


Bild 2. Wenn Sie durch den Nebel aus passiven Filtern, Entkopplungskondensatoren, Datenbussen und Versorgungsleitungen blicken, bleibt eine Schaltung mit nur vier Chips übrig!

trifft man mit der Lötbrücke JP1. Auf der gleichen Stiftleiste sind auch sieben zusätzliche I/O-Leitungen verfügbar, die für Funktionen wie PTT (Push-to-Talk) und Morse-Tonsignale vorgesehen sind. Sie können auch als I/O für I²S-Audio verwendet werden.
Die Lötbrücken JP2 und JP3 haben keine Funktion in der aktuellen FPGA-Firmware. K6 stellt die JTAG-Programmierschnittstelle des FPGAs und dessen Konfigurationsspeichers dar.

Topologie der Funkplatine

Mit einer geeigneten Funkplatine kann das FPGA/DSP-Board für den Funkbetrieb bei nahezu jeder Frequenz zwischen null und vielen GHz verwendet werden. Die typischen Anwendungen sind jedoch Kurzwelle (unter 30 MHz) und/oder das 2-m-Amateurband (144...146/148 MHz). Für den Kurzwellenempfang könnte eine einfache Funkplatine wie in **Bild 3** konstruiert werden. Das Antennensignal wird tiefpassgefiltert und verstärkt. Durch Mischen mit einem lokalen Oszillator-Signal von DAC A wird die entsprechende Frequenz auf 45 MHz konvertiert, bevor das Signal durch ein Quarzfilter geführt wird. Dieses Zwischenfrequenzsignal (ZF) wird vor einem zweiten Mischer verstärkt und dann mit einem zweiten LO-Signal (von DAC B) zu einer zweiten ZF bei 312,5 kHz (siehe unten) reduziert. Dieses zweite ZF-Signal wird dann vom ADC digitalisiert und in das FPGA geleitet, wo es verarbeitet und letztendlich zu einem Audiosignal demoduliert wird.
Im Vergleich zu vielen anderen SDR-Lösungen ist diese Topologie aus zwei Gründen vorteilhaft:

1. Prinzip des Doppel-Überlagerungsempfängers

Das Signal wird zu einer ersten, hochfrequenten ZF gemischt, wo es gefiltert wird. Dann wird es in eine niederfrequente zweite ZF umgewandelt und anschließend abgetastet. Der Vorteil ist, dass starke Signale auf nicht interessierenden Frequenzen durch das schmale Filter schon früh in der Kette herausgefiltert werden. Dies bedeutet eine hohe Selektivität oder einen weiten Dynamikbereich, so dass der Empfänger schwache Signale in Gegenwart von starken gut empfangen kann.
Die einfacheren SDR-Projekte verwenden oft das „Null-ZF"-Prinzip mit nur einem Mischer ohne Filter und digitalisieren einfach auf Audio-Frequenzen rund um den

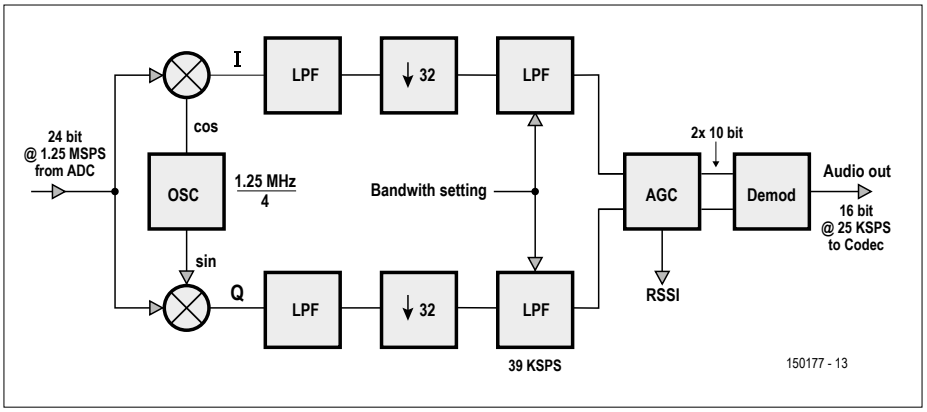


Bild 4. Die im FPGA implementierte Signalverarbeitung.

Träger. Der Ansatz ist aber nicht in der Lage, starke störende Sender wirksam zu unterdrücken, da die Mischprodukte unerwünschter Signale wahrscheinlich in das Audio-Band fallen, wo sie das Nutzsignal verzerren.
Originellere SDR-Lösungen tasten einfach einen großen Teil des Frequenzspektrums ab (direct sampling) und erledigen die Abstimmung in der digitalen Domäne. Diese Methode schafft interessante Möglichkeiten, weite Bereiche des Spektrums abzuhören, aber für eine gute Selektivität und/oder Dynamik werden sehr strenge Anforderungen an die Linearität des Eingangsverstärkers und die tatsächliche Auflösung des ADCs (effektive Bits) gestellt. Schnelle Abtastraten und hohe Auflösungen bedeuten aber in der Regel hohe Kosten und hohe Leistungsanforderungen. Das alles führt in der Praxis leider zu oft zu inakzeptablen Kompromissen.

2. Die zweite ZF ist niedrig, aber nicht null

Dies bedeutet, dass ein störender Sender innerhalb der Filterbandbreite von typischerweise 15 kHz keine Mischfrequenzen erzeugen kann, die das empfangene Nutzsignal stören. Diese Art von Mischprodukten wird zwar mitgesampelt, aber anschließend digital herausgefiltert, weil sie unterhalb des gewünschten Signals mit der Mittenfrequenz von 312,5 kHz liegt. Da diese Frequenz nicht zu hoch ist, ist eine Abtastrate im Bereich von 1...2 MSPS ausreichend, was auch preiswerte 24-Bit-ADCs problemlos schaffen. Die Auflösung von 24 Bit bedeutet einen digitalen Dynamikbereich von mehr als 100 dB und macht gleichzeitig eine Verstärkungsregelung in den analogen Schaltungsblöcken überflüssig, was wiederum die Empfängerschaltung stark vereinfacht.

Listing 1. VHDL-Implementation eines „Digital Down Converters" bei einem Viertel der Samplefrequenz.

```
if ns = 0 then
Ia(write_pointer) <= signed(Data_in);-- 1
Qa(write_pointer) <= to_signed(0,24);-- 0
ns := 1;
elsif ns = 1 then
Ia(write_pointer) <= to_signed(0,24);-- 0
Qa(write_pointer) <= signed(Data_in);-- 1
ns := 2;
elsif ns = 2 then
Ia(write_pointer) <= (not signed(Data_in)) + 1; -- -1
Qa(write_pointer) <= to_signed(0,24);-- 0
ns := 3;
elsif ns = 3 then
Ia(write_pointer) <= to_signed(0,24);-- 0
Qa(write_pointer) <= (not signed(Data_in)) + 1; -- -1
ns := 0;
end if;
```

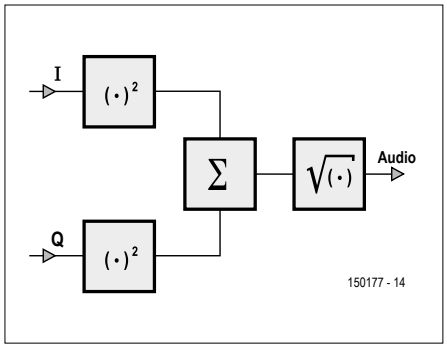


Bild 5. AM-Signaldemodulation.

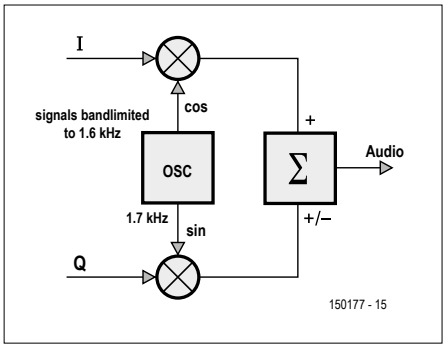


Bild 6. SSB- und CW-Signaldemodulation.

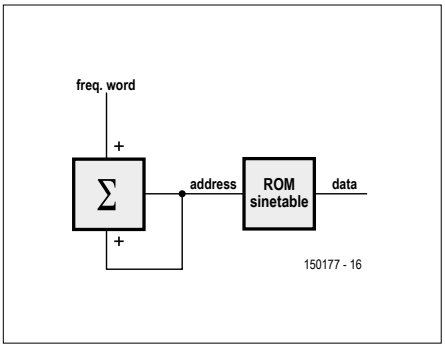


Bild 7. Prinzipielle Arbeitsweise der DDS. Der Block mit dem Sigma-Symbol bezeichnet den Phasendemodulator.

Digital Signal Processing – der Empfang

Es ist Aufgabe des FPGAs, das abgetastete Eingangssignal zu verarbeiten. **Bild 4** zeigt ein vereinfachtes Diagramm der Blöcke im FPGA.

Der ADC digitalisiert das auf 312,5 kHz zentrierte Eingangssignal mit einer Rate von 1,25 Millionen Samples pro Sekunde (MSPS). Mit 24 Bits pro Sample sind bei 1,25 MSPS 30 Mbits/s zu verarbeiten. Dies ist viel mehr als erforderlich, so dass in einem ersten digitalen Verarbeitungsschritt der Eingangs-Datenstrom heruntergetaktet wird (downsampling). Dazu wird das Signal mit zwei Sequenzen multipliziert und so bei 312,5 kHz in ein Quadratursignal reduziert, das bei 0 Hz zentriert ist. Die eine Sequenz [0,1,0, -1,0, ...] ist eine „Sinuswelle" bei genau einem Viertel der Abtastrate, die andere [1,0, -1,0,1, ...] ist identisch, aber um 90° verschoben, also ein Cosinus. Diese Signale werden dann zu Down-Sampling-Tiefpassfiltern geführt, die die Abtastraten um den Faktor 32 reduzieren, einfach dadurch, dass sie 31 von 32 Samples verwerfen, und so Ausgangssignale mit Abtastraten von rund 39 KSPS liefern. Diese beiden heruntergetakteten Signale enthalten nun jeweils Informationen von null bis etwa 10 kHz. Zusammen repräsentieren sie die Information, die innerhalb von ±10 kHz rund um 312,5 kHz des ursprünglich abgetasteten Eingangssignals lag.

Falls Sie es nicht erraten haben: Die zweite ZF wurde auf 312,5 kHz gelegt, weil dies ein Viertel der 1,25-MHz-Abtastrate ist. Dadurch ließ sich der Down-Konverter sehr einfach in VHDL implementieren, weil multiplizieren mit 1, 0 und -1 nicht besonders kompliziert ist (siehe **Listing 1**). Andere ZF-Frequenzen hätten eine Sinustabelle und 24-Bit-Multiplizierer erfordert.

Im nächsten Schritt werden beide Signale noch einmal gefiltert. Diesmal sind die Grenzfrequenzen für die gewählte Betriebsart einstellbar. Für AM liegt die Grenzfrequenz bei 5 kHz für 10 kHz Informationsbandbreite, für SSB-Telefonie (Single-Side-Band) ist sie auf etwa 1,6 kHz (3,2 kHz Bandbreite) eingestellt und für schmalbandiges CW (Continuous Wave, Morse Code) filtern sie bei nur wenigen hundert Hertz.

Bei zweimal 24 Bit Auflösung sind immer noch enorm viele Daten zu verarbeiten. Um die interessanten Informationen zu extrahieren (aka demodulieren), sind nicht all diese Bits nötig. Die Aufgabe des nächsten Blocks, die automatische Verstärkungsregelung (AGC), ist es daher, die Signalstärke kontinuierlich zu überwachen und das Signal so zu skalieren, dass nur zwei 10-Bit-Ströme ausgegeben werden, die dann demoduliert werden können. Der AGC-Block gibt auch das Signal für eine Empfangssignalstärke-Anzeige (Received Signal Strength Indication, RSSI) aus.

Dann geht es zum Demodulator. Für die AM-Signal-Demodulation bedeutet dies eine Extraktion der Amplitude durch das Gesetz des Pythagoras für zwei orthogonale Vektoren: $\sqrt{(I^2+Q^2)}$, siehe **Bild 5**. Für SSB (und CW) ist die Demodulation leider etwas komplizierter, so dass das Modell von Weaver verwendet werden musste [2][3]. Die Hauptidee (**Bild 6**) dabei ist, die Signalbandbreite in zwei Teile zu teilen und wieder mit einem lokal erzeugten niederfrequenten Quadratursignal in der Mitte des Bandes zu mischen. Die Wahl zwischen Upper Side Band (USB) oder Lower Side Band (LSB) ist dann nur eine Frage der Vorzeichen in einem abschließenden Addierer.

Das demodulierte Audiosignal durchläuft schließlich ein Tiefpassfilter, um jegliche Hochfrequenzanteile zu entfernen, die durch Verzerrungen im AGC oder im Demodulator verursacht wurden. Das Signal wird dann neu abgetastet, um es der Abtastrate des Audio-Codecs anzugleichen und dann über eine serielle Standard-PCM-Schnittstelle zu senden. Der Audio-Codec bietet eine programmierbare Verstärkung und kann ohne weiteres einen Kopfhörer oder auch einen kleinen Lautsprecher ansteuern.

Die Signale des lokalen Oszillators (LO) an den Ausgängen der beiden DACs werden durch zwei DDS-Blöcke (Direct Digital Synthesis) im FPGA erzeugt. Ein solcher Synthesizer besteht aus einem so genannten Phasenrad (phase accumulator register), dessen Wert bei jedem Taktzyklus des 120-MHz-Taktes des DACs (der von der internen PLL des FPGA abgeleitet wird) um einen festen Betrag erhöht wird. Diese Erhöhung bestimmt die Frequenz. Der akkumulierte Wert wird als Adresse in einer Sinustabelle verwendet und der an dieser Adresse gefundene Wert an den DAC gesendet (**Bild 7**). Das Frequenzwort für DAC A, der den abstimmbaren LO darstellt, ist 25 Bit breit. Dies entspricht einem Frequenzschritt von $120 \times 10^6/2^{25} = 3{,}57$ Hz. Darüber hinaus gibt es einen Mechanismus zur Feinabstimmung, der mit gebrochenen Zahlen arbeitet, so dass eine endgültige Auflösung von 0,44 Hz erreicht wird. Für den zweiten LO mit seiner festen Frequenz für die Umwandlung von der ersten zur zweiten ZF wird ein entsprechender DDS-Block mit einer niedrigeren Auflösung von etwa 38 Hz implementiert, um FPGA-Ressourcen zu sparen.

FPGA-Programmierung

Die kompilierte Firmware für dieses Projekt steht als „JTAG indirect configuration"-Datei (trx.jic) zur Verfügung. Sie konfiguriert zunächst das

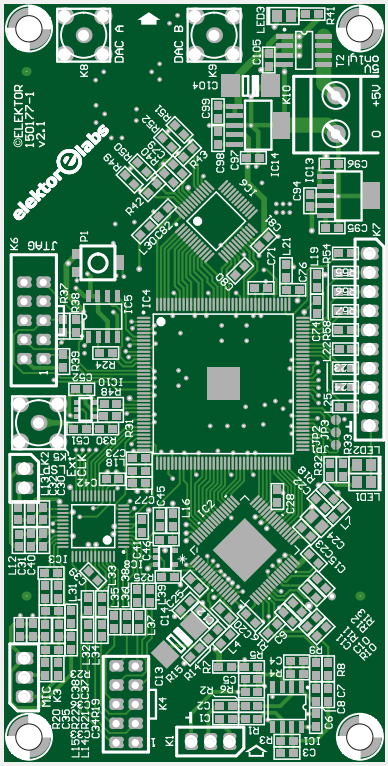
STÜCKLISTE

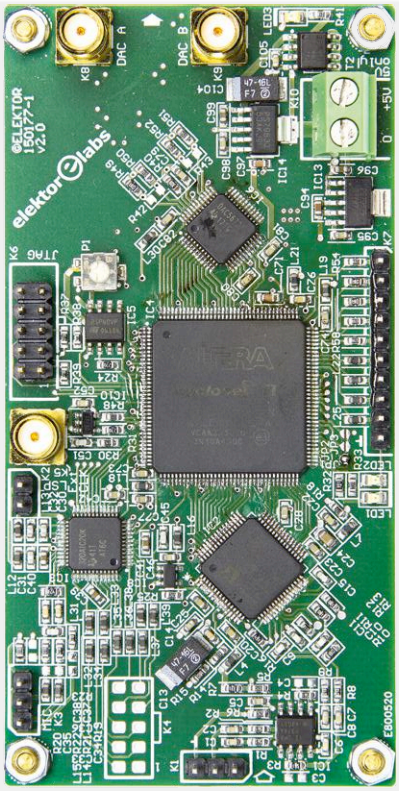
Widerstände:
alle 1%, 100 mW, 0603
R1,R2,R3,R4 = 560 Ω
R5,R6,R25,R28,R30,R34,R35,R36,R44...R47,R48,R53 = 10 k
R7,R32,R33 = 330 Ω
R8...R11,R19...R22,R37,R38,R39,R41 = 1 k
R12,R13 = 18 Ω
R14,R17,R18 = 10 Ω
R15,R49...R52,R54...R58 = 100 Ω
R16 = 160 k
R23 = 47 k
R24,R29,R31 = 27 Ω
R26 = 100 k
R27,R42,R43 = nicht bestückt
R40 = 2 k
P1 = 10-k-Trimmpoti, 4,5 mm, SMD

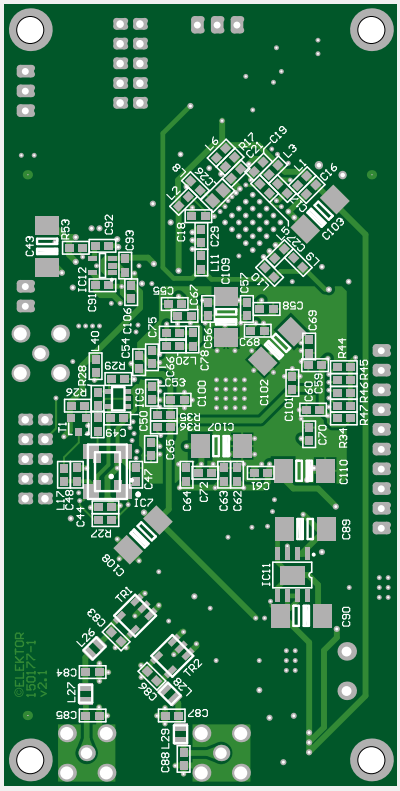
Kondensatoren:
alle 5%, 0603 (w.n.a.a)
C1,C2,C5...C8,C14,C16...C19,C20,C22,C23,C25,C27,C28,C29, C42,C44,C46,C47,C48,C50,C52,C53...C64,C66,C68,C70,C72...C82,C92,C95,C98,C105,C106 = 100 n
C3,C4,C33,C34,C35 = nicht bestückt
C9 = 5p6 ±0p25
C10,C11 = 27 p, 1%
C12,C21 = 10 n
C13,C43,C89,C90,C102,C103,C104,C107...C110 = 47 µ, 16 V, Tantal, 0Ω35, SMD-Gehäuse C
C15 = 33 p
C24,C26 = 10 µ, 6,3V, 0805
C30,C31,C32 = 10 p
C36 = 1 n
C37,C38 = 22 n
C39,C40,C41,C93,C96,C99 = 1 µ, X7R
C45,C49,C51 = 100 p
C83,C85,C86,C88 = 120 p
C84,C87 = 180 p
C65,C67,C69,C71,C91,C94,C97,C100,C101 = 2µ2, X7R

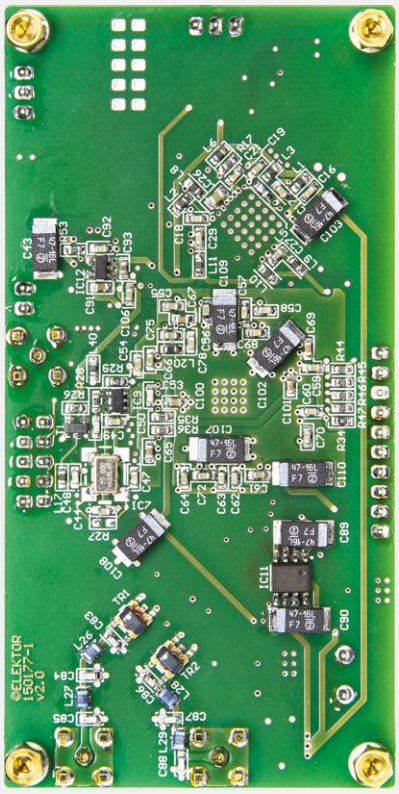
Induktivitäten:
L1...L4,L6...L25,L30...L40 = 1 kΩ @ 100 MHz, 200 mA, 0603
L5 = 15 nH, 5%, 170 mΩ, 700 mA, f_{res} 4 GHz
L26...L29 = 180 nH, 2%, 640 mΩ, 400 mA, 0805
TR1,TR2 = TC4...1WG2+

Halbleiter:
IC1 = OPA2374AIDG4
IC2 = AD7760BSVZ
IC3 = TLV320AIC20KIPFB
IC4 = EP4CE10E22C8N
IC5 = M25P40-VMN6PB
IC6 = DAC5672IPFB
IC7 = Quarzoszillator 20 MHz, einstellbar, 5×3,2 mm²
IC8,IC9,IC10 = 74AHC1GU04W5-7
IC11 = LP3891EMR-1.2/NOPB
IC12 = TPS73018DBVT









IC13 = TPS79625DCQG4
IC14 = TPS79633DCQ
LED1,LED2,LED3 = grün, 0805
T1 = PMV65XP
T2 = IRF9321PBF

Außerdem:
K1,K3 = 3-polige Stiftleiste, Raster 0,1”
K2 = 2-polige Stiftleiste, Raster 0,1”
K4 = 14-polige Stiftleiste, Raster 0,1”
K5,K8,K9 = gerade SMA-Buchse, 50 Ω
K6 = 2x5-polige Stiftleiste, Raster 0,1”
K7 = 10-polige Stiftleiste, Raster 0,1”
K10 = 2-polige Platine-Anschlussklemme, Raster 0,2”
Platine 150177-1

Bild 8. Der Prototyp des Autors im USB-Betrieb bei 144 MHz.

Weblinks

[1] www.elektormagazine.de/150177

[2] Donald K. Weaver Jr., „A Third Method of Generation and Detection of Single-Sideband Signals“, www.h4.dion.ne.jp/~ja5fp/weaver.pdf

[3] Daniel Uppström, „Weavers metod för SSB“, ESR Resonans 2/2014, http://resonans.esr.se/ESR_Resonans_2014_2.pdf (Schwedisch)

[4] https://www.elektormagazine.de/labs/fpga-dsp-radio-for-narrow-band-communications-150177-i

[5] https://github.com/ast/dspsdr

FPGA als Bridge und programmiert dann den Konfigurationsspeicher über das FPGA. Dies geschieht physikalisch mit einem USB-Blaster-Adapter zwischen der USB-Schnittstelle des PCs und der JTAG-Schnittstelle auf der Platine. Solche Adapter findet man zu geringen Preisen auf den verschiedenen Shopping-Webseiten.
Die Software auf der PC-Seite ist **Quartus Lite**; sie kann von der Intel/Altera-Website kostenlos heruntergeladen werden. Man benötigt nicht einmal die komplette Entwicklungsumgebung, die für die Programmierung benötigten Applikationen reichen völlig. Die Quartus Suite ist sowohl für Windows als auch für Linux verfügbar.

Control-Interface

Beim Einschalten wird zunächst die Firmware aus dem Konfigurationsspeicher geladen und in das FPGA programmiert. Dann muss das FPGA über I²C oder den seriellen Port initialisiert und angesteuert werden. Das FPGA verhält sich ähnlich wie ein typisches IC mit ein paar Registern, die gesetzt werden müssen. Bei jeder Datenübertragung werden fünf Bytes an das FPGA gesendet, von denen die ersten beiden Bits die Registeradresse setzen und die restlichen 38 Bits die Konfigurationsdaten enthalten.
Der Controller fragt einige Statussignale und den Wert der Empfangssignalstärke (RSSI) zyklisch einige Male pro Sekunde ab und gibt sie für die Benutzerschnittstelle aus.
Eine detaillierte Beschreibung des Control-Interfaces mit vollständiger Registermap steht zusammen mit der Firmware und den Sourcedateien in einem GIT-Repository zur Verfügung, auf das über [1], [4] und [5] zugegriffen werden kann.
In einem Standalone-Funkgerät kommt als Steuerung üblicherweise ein kleines Mikrocontroller-Board mit Display und Tasten zum Einsatz, zum Beispiel der Elektor-Platino. Ein solches Board soll auch in einem nachfolgenden Artikel beschrieben werden. Für Experimente (oder auch, wenn man einen Monitor und Maus/Tastatur verwenden möchte), kann auch ein Raspberry Pi als flexibles Control-Interface eingesetzt werden. Für den RPi (oder einen anderen Linux-Computer) gibt es ein Python-Applet, das mit dem GTK-Grafik-Framework entwickelt wurde und das die Steuerelemente in einem GUI zugänglich macht.
Standardmäßig nutzt dieses Applet die serielle Schnittstelle der FPGA-Platine. Es besitzt auch einen (experimentellen) Socket-Modus für eine mögliche Fernbedienung. Die Idee ist dabei, einen Server auf dem Raspberry Pi am Ort des Funkgeräts zu starten und dann eine Steuerung mit einem (weit) entfernten Computer zu ermöglichen. Bei den vielen künstlichen Störsignalen, wie man sie in städtischen Bereichen vorfindet, und den Beschränkungen, die den Amateurfunkern in Bezug auf die Antennengröße auferlegt sind, wird die Option einer Fernbedienung für Funkamateure und Kurzwellenhörer zunehmend interessant.
Man könnte in einer fernbedienten Anwendung auch das Audiosignal vom FPGA direkt zu den PCM/I²S-Ports des Raspberry Pi leiten, doch dies muss noch ausprobiert werden.
Laden Sie also das Applet herunter, installieren Sie die notwendigen Pakete und verbinden Sie den Raspberry Pi mit dem FPGA-Board, wie es in einem Dokument beschrieben wird, das zum Download-Paket gehört.

Vorläufiges Fazit

Das hier präsentierte Projekt ist ein leistungsfähiger Baustein für Funkamateure und alle, die eigene Funkgeräte ohne Kompromisse bei der Leistungsfähigkeit bauen wollen. Es bleibt zudem zu hoffen, dass dieses Projekt die mehr „digital orientierten“ Leser dazu anregen wird, in die Welt des Funks und der Signalverarbeitung einzutauchen.
Das Projekt bietet viel Raum für Verbesserungen und Ergänzungen. Ein zukünftiger Artikel (vielleicht schon in der nächsten Elektor-Ausgabe) wird dann ein Funk-Board vorstellen, bei ausreichendem Interesse kommen auch weiter fortgeschrittene Modelle hinzu.
Der VHDL-Code für das FPGA steht unter [5] zur Verfügung. Ergänzungen und Verbesserungen sind stets willkommen!
Schließlich bleibt anzumerken, dass das Board „generisch“ genug ist, um auch als allgemeine Entwicklungsplattform für FPGA- und DSP-Anwendungen eingesetzt zu werden. Bitte besuchen Sie die Webseite des Projekts, um über Updates und Ergänzungen auf dem neusten Stand zu bleiben. ⏮

(150177)

EINKAUFSZETTEL

→ **150177-1:** Leerplatine für das FPGA-DSP-Board

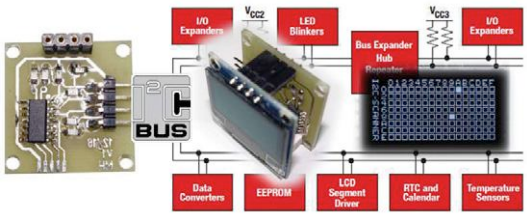
→ **150177-91:** Fertig bestückte FPGA-DSP-Platine

Elektor Labs Pipeline

Bei einem großen Teil der Elektronik geht es darum, etwas zu messen. Dieses Mal präsentieren wir Ihnen einige Selbstbauprojekte, mit denen Sie Hitze, Abfall, Zeit und I²C-Adressen messen können.

I²C-Bus-Scanner mit OLED-Display

Der Inter-Integrated-Circuits-Bus, kurz I²C-Bus, wurde von Philips in den 1980er Jahren entwickelt, um einen Mikroprozessor oder Mikrocontroller einfach mit anderen integrierten Schaltungen (meist in Fernsehgeräten von Philips und von Tochterfirmen) zu verbinden. Doch nach und nach wurde der Bus populär, so dass heutzutage Bauteile mit I²C-Schnittstelle allgegenwärtig sind. Weil der Bus überall eingesetzt wird, ist ein Werkzeug notwendig, das überprüft, ob er richtig funktioniert. Der hier vorgeschlagene Scanner hilft bei der schnellen Analyse und überprüft, ob die Busteilnehmer tatsächlich noch am Leben sind.



https://goo.gl/4ZUy79

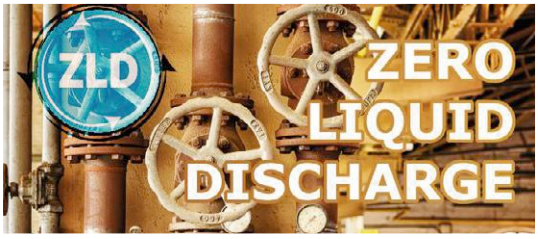
Candle2light — ein Lichtausbeute-Verstärker

Vor der Erfindung der elektrischen Beleuchtung im frühen 20. Jahrhundert wurden meist Kerzen und Öllampen zur Beleuchtung eingesetzt. Doch auch eine moderne Kerze, die etwa 80 W Wärmeenergie produziert, hat eine fast hundertmal niedrigere Lichtausbeute als eine Glühlampe. Und bei Teelichtern ist es noch schlimmer. Mit einem Peltier-Element ist es möglich, die Wärme eines Teelichts in Elektrizität umzuwandeln und damit eine hocheffiziente LED-Leselampe zu betreiben. So können Sie Ihr Lieblingsbuch bei elektrischem Kerzenlicht lesen!

https://goo.gl/CIyKwl

„Zero Liquid Discharge"-Überwachung per LoRaWAN

„Zero Liquid Discharge" (ZLD) ist ein Konzept hin zu einer nicht umweltschädlichen Industrieanlage. Es bedeutet, dass eine Industrieanlage Wasser zwar verwenden kann, aber kein Abwasser erzeugt. ZLD-Anlagen produzieren feste Abfälle. Dieses Projekt ermöglicht eine kostengünstige, solarbetriebene, drahtlose Überwachung von weitläufigen Fabrikanlagen, die mehrere Abflüsse besitzen. Ein Ultraschall-Transceiver, ein Arduino und ein LoRaWAN-Modul ist alles, was man an einem Abfluss benötigt. Plus einen Laptop mit Google Maps im Büro.

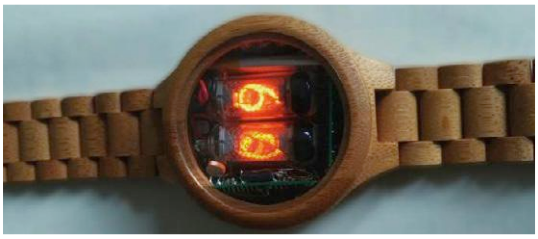


https://goo.gl/KyU7G1

So Retro, so stylish: die hölzerne Nixie-Armbanduhr

Seit vielen Jahren bin ich stolzer Besitzer eines stilvollen, glänzenden, hölzernen Attaché-Koffers, den ich von einer Reise nach Tansania mit nach Hause gebracht habe. Ich benutze ihn aber nicht, denn leider bin ich dazu nicht „stylish" genug. Vielleicht verbessert sich mein Stil, wenn ich diese schöne hölzerne Uhr mit ihrem Retro-Nixie-Röhren-Display baue? Vielleicht sollte ich auch hölzerne Brillen tragen? Oder vielleicht sollte ich einfach aufhören, Holzschuhe zu tragen? ⏮

(160401)



https://goo.gl/lUMZRU

MicroPython und das Pyboard

Von der blinkenden LED zur … blinkenden LED am Webserver

Von **Clemens Valens** (Elektor-Labor)

Seit ihrer Geburt im Jahre 1991 hat die Programmiersprache Python eine weite Verbreitung erlangt. Heute ist sie sogar die bevorzugte Sprache vieler Programmierer. Obwohl für PC-Anwendungen entwickelt, wurde Python vor einigen Jahren auch auf eingebettete Systeme portiert. Unter der Bezeichnung MicroPython läuft eine abgespeckte Version auf Plattformen wie dem BBC micro:bit und dem ESP8266. Und na klar, sie läuft natürlich auch auf dem offiziellen Pyboard.

Das „Micro Python"-Projekt (damals getrennt geschrieben) war einschließlich des Pyboards Ende des Jahres 2013 auf Kickstarter und sammelte das Sechsfache des ursprünglichen Finanzierungsziels von 15.000 £ ein. Diesen Erfolg konnte das Team durch die Unterstützung des WLAN-Moduls CC3000, des Ethernet-Moduls WIZ820io und des Low-Power-Funkmoduls NRF24L01+ erringen. Jetzt, fast vier Jahre später, wollen wir sehen, was daraus geworden ist. Beginnen wir mit dem Pyboard und µPy.

Erste Schritte mit dem Pyboard

Vor Ihnen sollte ein Pyboard v1.1, ein Micro-USB-Kabel und ein Computer (ich nutze einen Windows-10-Laptop) liegen. Wenn Sie die Platine an den Computer anschließen, erscheint sofort ein Fenster mit dem Inhalt des Laufwerks „E" und der Bezeichnung PYBFLASH. Wenn eine SD-Karte eingesteckt ist, wird stattdessen deren Inhalt angezeigt. Das Betriebssystem sollte auch einen seriellen Port finden. Dann starten Sie ein serielles Terminal wie TeraTerm, verbinden es mit dem Pyboard-Port und drücken eine Taste. Nun sollte man den Python-Prompt sehen (**Bild 1**):

```
MicroPython v1.8.2 on 2016-07-13;
PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>>
>>>
```

Das ging fix, nicht wahr? Weniger als eine Minute, nachdem man das Board an den Computer gestöpselt hat, ist man auch schon im Geschäft. In dieser Minute ist man sonst meistens mit der Maus in

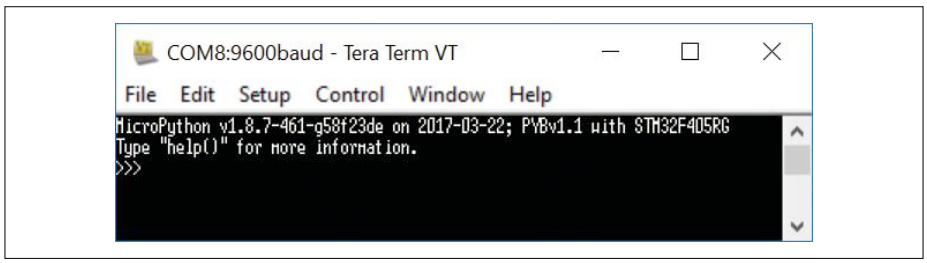


Bild 1. Die REPL-Programmierumgebung.

irgendwelchen Menüs des Betriebssystems unterwegs.

REPL

Was Sie jetzt sehen, ist schon eine einfache, interaktive Programmierumgebung, die einzelne Ausdrücke annimmt, sie auswertet und das Ergebnis zurückgibt. Eine solche Umgebung wird Read-Evaluate-Print-Loop oder kurz REPL genannt. Diese Abkürzung sollte man sich merken, denn man stößt bei der Arbeit mit Python öfter darauf. Wer sich einst mit BASIC beschäftigt hat, wird diese Art von Schnittstelle kennen, obwohl sie damals sicher nicht als REPL bezeichnet wurde. Python ist wie BASIC eine interpretierte Programmiersprache. Python-Programme können wie BASIC-Programme kompiliert werden, damit sie schneller laufen. Tatsächlich, Python ähnelt BASIC sehr.

Der nächste Schritt: Eine LED blinkt

Wie der Python-Prompt schon vorgeschlagen hat, geben Sie nun „help()“ ein und drücken die Eingabetaste. Es folgt eine Liste möglicher Befehle, mit denen man direkt loslegen kann (**Bild 2**). Ein genauer Blick auf die Liste zeigt, was mit dem Board so alles möglich ist. Das Board mit dem ARM-Cortex-M4F-Mikrocontroller STM32F405RG bietet offensichtlich eine Echtzeituhr (RTC), einen Analog-Digital-Wandler (ADC), einen Digital-Analog-Wandler (DAC), vier LEDs, Schalter, einen Beschleunigungssensor, einen Zufallszahlengenerator (RNG), I²C, SPI, UART sowie I/O-Pins und unterstützt zu allem Überfluss auch noch Servos. Einige Funktionen, die diese Peripherie steuern, sind ebenfalls aufgeführt. Und wenn man das Board anschaut, sieht man natürlich auch den SD-Karten-Sockel. Geben Sie am Prompt den folgenden Befehl ein und drücken Sie <Enter>:

```
>>> pyb.LED(1).on()
```

Jetzt sehen Sie, dass eine rote LED neben dem USR-Taster aufleuchtet. Wiederholen Sie den Befehl, aber ersetzen Sie die „1“ durch eine „2“ (benutzen Sie die Pfeiltasten, um im Terminal zum vorherigen Befehl zurückzukehren und ihn zu bearbeiten). Jetzt leuchtet eine grüne LED neben der roten. LED No. 3 ist gelb, Nummer 4 blau. Mit Ausnahme der gelben leuchten die LEDs unglaublich hell, schauen Sie besser nicht direkt hin! Lassen Sie uns also mit dem gedämpften Licht der gelben LED 3 fortfahren und uns die anderen grellen Exemplare ersparen. Damit eine LED blinkt, benötigt man immer eine Schleife, zum Beispiel hier eine `while`–Schleife. Zu einer Endlos-Schleife gehört stets auch eine Bedingung, die immer wahr ist/wird, etwa die Boolesche Konstante `True`. Geben Sie den Befehl ein und vergessen Sie nicht den Doppelpunkt „:“ am Ende.

```
>>> while True:
```

Als ersten Befehl geben Sie den Toggle-LED-Befehl ein. Das Terminal beginnt mit dem Einrücken des Codes, wie es in Python obligatorisch ist, hier angezeigt durch drei Punkte. Geben Sie auch noch eine Verzögerung ein, um den Blinktakt einzustellen, zum Beispiel 250 ms. Dann sollte im Terminal ungefähr so etwas stehen:

```
>>> while True:
...     pyb.LED(3).toggle()
...     pyb.delay(250)
...
```

Um diese Schleife auszuführen, muss zuerst der Einrück-Modus verlassen werden. Dazu drückt man in der nächsten Zeile die Backspace- und dann die Ein-

Eigenschaften

- Programmierung in MicroPython
- Einfacher Webserver
- 100% Arduino-frei

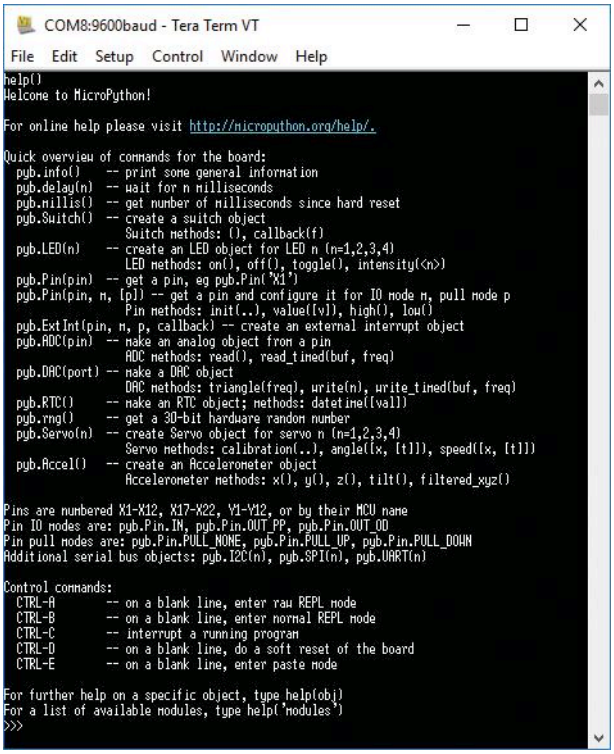


Bild 2. MicroPython enthält eine umfangreiche Hilfe.

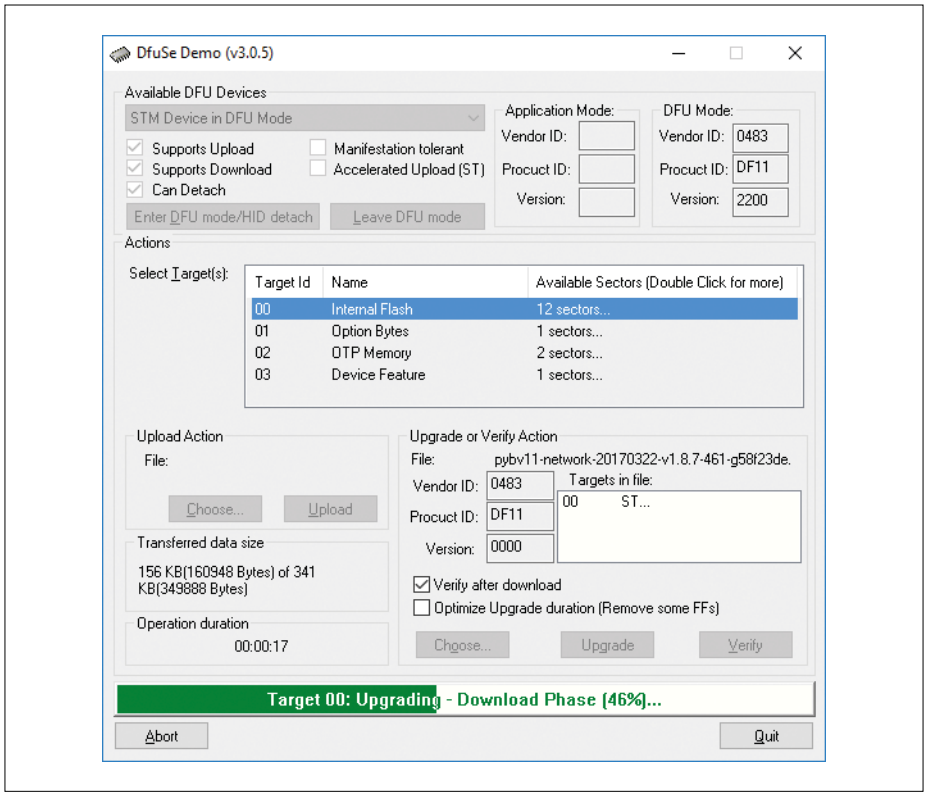


Bild 3. DfuSe schreibt die neue Firmware auf das Pyboard.

gabe-Taste. LED3 sollte nun mit der eingetragenen Frequenz blinken.

Wie Sie vielleicht bemerken, akzeptiert das Terminal keine Befehle, während die Schleife läuft. Um aus der Schleife hinauszukommen und zum Eingabe-Prompt zurückzukehren, drücken Sie Strg-C (Unterbrechen eines laufenden Programms). Aufgepasst: Alle Befehle, die man irrtümlich oder versehentlich während des Programmablaufs eingegeben hat, werden jetzt ausgeführt!

Beschleunigungsaufnehmer

Kommen wir zu einem kurzen Experiment, das überprüft, ob der Beschleunigungsaufnehmer des Boards richtig funktioniert.

```
>>> while True:
...     print(pyb.Accel().x(),pyb.Accel().y(),pyb.Accel().z())
```

Die Ausgabe sollte eine endlose Liste der drei Werte x, y und z sein. Drehen, kippen, schütteln Sie das Board, dann werden Sie sehen, wie sich die Werte ändern.

```
-1 -2 21
-1 -2 20
-2 -1 20
…
```

Drücken Sie wieder die Taste Strg-C, um zum Eingabe-Prompt zurückzukehren. Der Test ist sehr aufschlussreich, zeigt er doch, wie schnell oder langsam eine scheinbar simple Schleife ausgeführt wird. Alles, was sie tut ist, den Sensor auszulesen und drei ganzzahlige Werte auszugeben. Ich habe etwa drei Zeilen pro Sekunde gemessen, was tatsächlich nicht sehr schnell ist.
Der Grund dafür ist, dass das Programm on-the-fly interpretiert wird, anstatt dass es zuvor in Maschinen-Anweisungen umgewandelt (kompiliert) wurde. Obwohl es nur eine Zeile Code (abgesehen von der While-Anweisung) zu sein scheint, handelt es sich in Wirklichkeit um einen sehr komplexen Befehl. Es ist zwar möglich, das Programm durch so genannte „frozen modules“ zu beschleunigen, aber dies würde eine Neukompilierung der kompletten Firmware für das Pyboard erfordern. Und so etwas würde in diesem Artikel zu weit führen.

Mein erstes Programm

Wenn Sie eine Weile mit REPL herumgespielt haben, können Sie sich eigentlich an Ihr erstes echtes eigenes Programm wagen. Alles, was Sie dazu benötigen, ist ein guter Texteditor.
Das Pyboard verhält sich wie ein Flash-Laufwerk. Wenn Sie es in einem Datei-Viewer öffnen, sehen Sie die Python-Datei main.py, die nach einem Reset (hart mit der RST-Taste oder soft mit dem Ctrl-D-Befehl) ausgeführt wird. Öffnen Sie diese Datei im Texteditor, fügen Sie Ihre Befehle hinzu und speichern Sie sie wieder. Wenn Sie nun das Pyboard zurücksetzen, startet das Ihr neues Programm.

Auswege aus der Konfusion

Fürchten Sie sich nicht davor, kreativ zu werden, Versuch und Irrtum sind der Weg zum Erfolg! Wenn Sie alles im Pyboard so durcheinander gebracht haben, dass Sie kein Licht mehr am Ende des Tunnels sehen, gibt es zwei Rettungsstrategien: Sicherer Modus und Factory-Reset. Im Sicheren Modus werden die Dateien `boot.py` und `main.py` nicht ausgeführt, so dass man Zugriff auf das Dateisystem erhält (das USB-Laufwerk sollte im Explorer erscheinen). Jetzt können Sie die Dateien bearbeiten, um die Probleme zu beseitigen. Um in den Safe-Modus zu kommen, drückt man die USR-Taste, hält sie gedrückt und tippt kurz auf die RST-Taste. Beobachten Sie die LEDs: Wenn nur noch die gelbe LED leuchtet, lassen Sie auch die USR-Taste los. Das Pyboard befindet sich jetzt im Safe-Modus.
Der Factory-Reset geht einen Schritt weiter. Es löscht alle Dateien im internen Pyboard-Speicher (nicht auf der SD-Karte) und bringt die Dateien `boot.py`, `main.py`, `README.txt` und `pybcdc.inf` wieder in ihren ursprünglichen Zustand. Um einen Factory-Reset auszuführen, geht man wie beim Safe-Modus vor, doch wenn die LEDs ihren kleinen Tanz aufführen, lassen Sie die USR-Taste los, wenn sowohl die gelbe als auch die grüne LED leuchten. Die gelbe LED blinkt dann drei Mal schnell und schon sind Sie da, wo Sie angefangen haben (gehe zurück auf Los!).
Wenn diese beiden Methoden nicht helfen, haben Sie einen letzten Ausweg: Umprogrammieren der Firmware. Dazu müssen Sie das Board in den Modus „Device Firmware Update“, DFU) bringen, indem Sie den Pin P1 mit 3V3 (etwa mit einer Büroklammer, ich habe bei mir einen Jumper verwendet) verbinden und dann die RST-Taste drücken. Wenn man zum ersten Mal zu diesem Strohhalm greift, muss man zunächst einen DFU-Treiber installieren. Hat Windows dies nicht ohnehin schon erledigt, muss man ein wenig nachhelfen. Laden Sie das Paket STSW-STM32080 (oder DfuSe

Demo) von der STMicroelectronics-Website herunter und installieren Sie es. Zeigen Sie Windows den rechten Pfad (<Installationspfad>\STMicroelectronics\Software\DfuSe v3.0.5\Bin\Driver\) und installieren Sie den Treiber. Wenn alles gut gelaufen ist, listet DfuSe-Demo nun das Board auf (**Bild 3**).
Vorkompilierte Firmware finden Sie auf der Website von µPy im Download-Bereich. Wählen Sie eine Firmware, die Ihrem Board entspricht. Im Bereich „Upgrade or Verify Action“ (nicht: „Upload Action“) klicken Sie auf „Choose“, navigieren zu der DFU-Datei, die Sie installieren wollen, und klicken dann auf „Upgrade“ (nicht: „Upload“). Nun schließen Sie DfuSe, entfernen den Jumper und drücken RST. Eventuell muss noch ein Factory-Reset durchgeführt werden, um das Flash-Dateisystem wiederherzustellen. In meinem Fall war es notwendig, Windows das Pyboard-Laufwerk nach dem Firmware-Update „reparieren“ zu lassen.
DFU ist auch unter Linux und Mac OS mit dfu-util oder pydfu möglich.

Auch andere machen Fehler

Wenn man damit anfängt, µPy-Programme zu schreiben, wird man natürlich eine Reihe von Fehlern machen und Irrtümer begehen. Einer der häufigsten - frustrierendsten - Fehler ist es, zu vergessen, die Tabulatortaste zu benutzen. Oder zu denken, man hat es getan, hat es aber nicht. Oder man tut es und sollte es nicht tun. Denken Sie daran, dass Python immer diese Einrückung erfordert und dafür nur Tab-Zeichen akzeptiert! Stellen Sie daher sicher, dass Ihr Texteditor Tabs speichert, denn es gibt viele, die das gerade nicht tun und Tabs durch Leerzeichen ersetzen. Am besten Sie nutzen den besten aller Freeware-Texteditoren Notepad++, der auch für Python eine Syntax-Hervorhebung kennt und Tabs als Tabs speichert.

Neben den Problemen mit dem Einrücken kann man auch Schwierigkeiten mit der Kompatibilität bekommen. MicroPython ist zwar ein Teil von Python, aber nicht Python. Auf der µPy-Website finden Sie eine Liste mit Unterschieden zwischen CPython (der „echten“ Python-Referenzimplementation) und µPy. Wenn also eines Tages ein merkwürdiger Fehler auftritt, wenn der Code nicht funktioniert, obwohl er dies eigentlich sollte, dann konsultieren Sie diese Liste!

Aufbau eines einfachen Webservers

Da die Kickstarter-Kampagne versprochen hatte, dass das Ethernet-Modul WIZ810io unterstützt würde und ich zufällig ein paar dieser kleinen Platinchen auf dem Labortisch hatte, beschloss ich, einen einfachen Webserver einzurichten (**Bild 4**). Doch damit wurden die Dinge kompliziert und dunkle Wolken zogen am Erkenntnishorizont auf. Zuerst wurde natürlich Google bemüht in der Hoffnung, ein alles erklärendes Beispiel zu finden, doch vergeblich. Ich habe zwar den Code für einen MicroPython-Webserver gefunden [4], aber er hat nicht auf meinem Pyboard funktioniert. Bevor Sie jetzt anfangen zu protestieren, ja, ich habe das Board zuvor mit einer speziellen Firmware mit dem Netzwerk-Treiber (`pybv11-network-20170322-v1.8.7-461-g58f23de.dfu`) neu programmiert.

Ich hatte den wirklich ganz einfachen Webserver-Code in `main.py` kopiert, aber er wollte nicht funktionieren. Aber wie kann man den Code debuggen? Ich weiß nicht, welchen Lösungsweg Python empfiehlt, vielleicht Debug-Anweisungen in das Programm einfügen oder das Programm zeilenweise in REPL kopieren. Ich habe, obwohl es etwas langweilig war, die zweite Methode gewählt. Dies zeigte mir schnell, dass der Aufruf der `Sendall`-Methode meines Socket-Objekts ungültig war (lesen Sie den Code unter [4], um zu verstehen). Es gibt nämlich gar keine solche Methode! Das ist ziemlich merkwürdig, denn die Socket-Dokumentation auf der MicroPython-Website erwähnt sie deutlich. Dies führte mich zum Quellcode von MicroPython auf der GitHub-Repository. Nach erheblicher Sucherei enthüllte ein genauerer Blick auf die Netzwerk-Treiber, dass der WIZnet-Treiber keine `Sendall`-Methode, ja, überhaupt keine Standard-Socket-Methoden kennt, die in MicroPython funktionieren oder sogar verwendet werden müssen. Stell' sich das einer vor, jemand hat bei der MicroPython Corp. Inc. Ltd. und Co. KG nicht alles gründlich getestet! Ok, das W7200-basierte WIZ810io-Modul ist ja auch nicht mehr das frischeste...
Im Nu war `sendall` durch `send` ersetzt, aber der Webserver wollte immer noch nicht arbeiten. Was fehlte, war ein `content-length`-Feld im Header der HTTP-Seite, die zu meinem Webserver gesendet wird. Nachdem ich auch

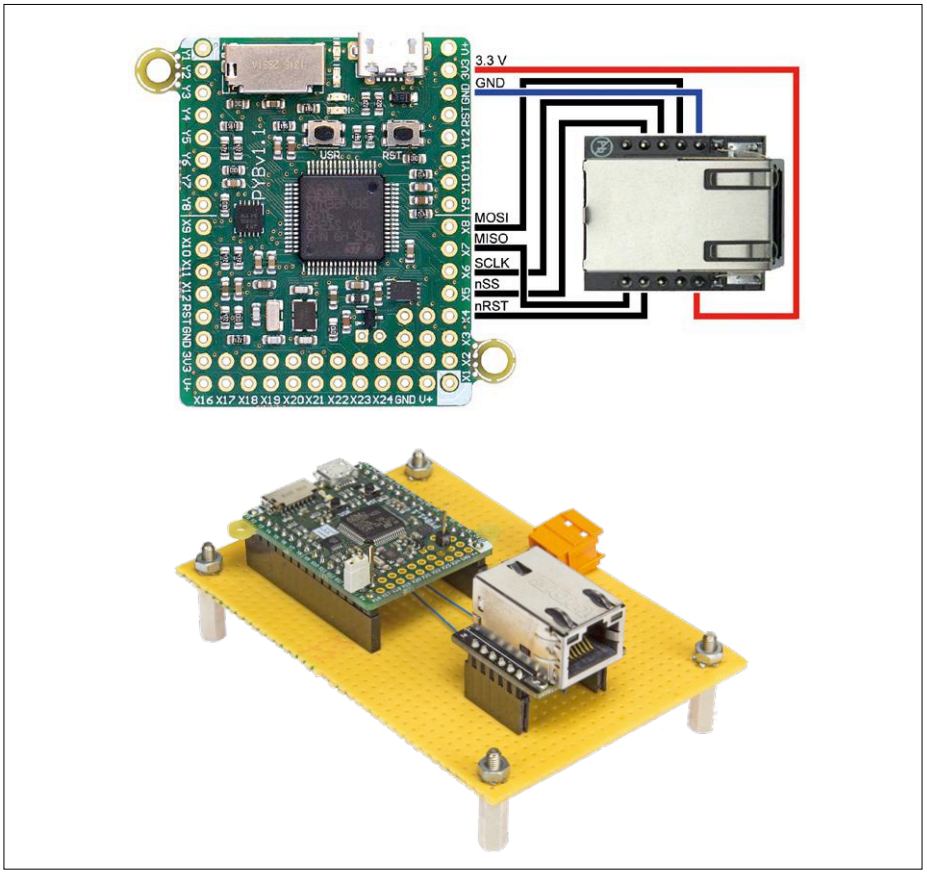


Bild 4. MicroPython-Webserver mit PYBv1.1-Pyboard und dem Ethernet-Modul WIZ810io von WIZnet.

dies hinzugefügt hatte, erschien meine MicroPython-Webseite endlich in meinem Browser (**Listing 1**).
Alles gut? Von wegen! Als ich den Code in `main.py` eingetragen hatte, funktionierte er, doch als ich ein paar Verbesserungen hinzufügte, ging etwas fürchterlich schief. Um es kurz zu machen: Das Dateisystem auf meinem Pyboard wurde irgendwie beschädigt und der Inhalt der Webseite war nur noch Müll. Auf die Datei `main.py` konnte ich nicht mehr zugreifen und ihre Größe war auf null geschrumpft. Nach Wiederherstellung des Pyboards mit den oben beschriebenen Methoden konnte ich mein kleines Programm neu installieren, um den Webserver wieder in Betrieb zu nehmen.
Diese Erfahrung lehrte mich, in REPL zu experimentieren, anstatt eine Datei zu oft auf dem Pyboard zu speichern. Das Dateisystem scheint ziemlich fragil zu sein. Für ein komfortables Copy-Paste-Arbeiten in REPL sollte man ein serielles Terminal wie TeraTerm benutzen, das mehrzeiliges Einfügen unterstützt. Damit lassen sich Codeblöcke bis dorthin einfügen, wo ein manueller Backspace erforderlich ist, um die Einrückbreite zu reduzieren (zum Beispiel am Ende einer `while`-Schleife oder `if-else`-Anweisung).

Zugabe, Zugabe!

Auf der MicroPython-Website finden sich schöne Tutorials, die zeigen, wie man die Peripherie des Pyboards benutzt. Man sollte sie durchlesen, das erspart viel Kopfzerbrechen und viel Zeit. Die Tutorials behandeln auch einige fortgeschrittene Techniken wie die Kombination von Assembler und µPy. Diese Tutorials beziehen sich allerdings ausschließlich auf das Pyboard und müssen nicht 100%-ig mit anderen Boards funktionieren.

MicroPython und das Pyboard sind eine spaßige und kostengünstige Methode, sich in die Embedded-Programmierung einzuarbeiten und dabei auch noch Python zu lernen. Alles, was die Kickstarter-Kampagne versprach, wurde erfüllt, auch die weiter gesteckten Ziele. Die MicroPython-Website mit ihren netten Tutorials ist nicht immer aktuell, also sollten Sie im Zweifelsfall die Projektseiten bei GitHub konsultieren! ◀

(160428)

Listing 1. Ein einfacher Webserver für das Pyboard.

```
import os
import network
import socket

# Adapt these for your network.
my_ip = '192.168.2.33'
subnet_mask = '255.255.255.0'
gateway = '192.168.2.1'
dns = '8.8.8.8' # dunno what to put here, 8.8.8.8 is WIZnet example.
# The one and only page we serve.
page_name = 'page1.htm'

# Initialize network interface.
nic = network.WIZNET5K(pyb.SPI(1),pyb.Pin.board.X5,pyb.Pin.board.X4)
nic.ifconfig((my_ip,subnet_mask,gateway,dns))
print(nic.ifconfig())
# Launch server (2x Ctrl-C gets you back into REPL mode).
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind(('',80))
s.listen(0)
while True:
  conn, addr = s.accept() # addr is not used but required.
  request = conn.recv(1024) # receive HTTP request.
  # Look for the string "Val=".
  val_begin = str(request).find('Val=')
  if val_begin>0:
    # Found string "Val=", now extract value.
    pyb.LED(1).on()
    val_end = str(request).find(' ',val_begin)
    v = str(request)[val_begin+4:val_end]
    print("Val =",v)
    conn.send(v)
    # LED3 off if v<50, on otherwise.
    if int(v)<50:
      pyb.LED(3).off()
    else:
      pyb.LED(3).on()
  else:
    # String "Val=" not found, serve web page.
    pyb.LED(2).on()
    # Send HTTP header.
    conn.send('HTTP/1.1 200 OK\nConnection: close\nServer: pyboard\
              nContent-Type: text/html\n')
    conn.send('Content-Length: ')
    # We have to insert the size of the data we are going to send.
    conn.send(str(os.stat(page_name)[6]))
    # Unreadable way of getting file size.
    # Close HTTP header.
    conn.send('\n\n')
    # Send web page.
    f = open(page_name,'r')
    conn.send(f.read())
  conn.close()
  pyb.LED(1).off()
  pyb.LED(2).off()
```

Weblinks

[1] www.elektormagazine.de/160428
[2] http://micropython.org/
[3] https://github.com/micropython
[4] https://github.com/RinusW/WiPy/tree/master/AiCWebserver

EINKAUFSZETTEL

→ **Elektor-Shop 17931:**
MicroPython pyboard v1.1

→ **Elektor-Shop 18023:**
Python for microcontrollers: getting started with MicroPython

(Fast) alles was Sie schon immer wissen wollten über ...

Bauteile in der Raumfahrt

Antworten von **Jaime Estela** (Spectrum Aerospace Group)

In Zeiten, in denen kleine Firmen und sogar Start-ups sich einen eigenen Satelliten leisten können, lohnt es sich, Fragen zu Elektronik im Weltraum zu stellen.

Q *Schädigen Strahlungseffekte elektronische Bauteile im All?*

A Im All werden alle elektronischen Bauteile von der Strahlung beschädigt. Die Robustheit der Bauteile hängt von vielen Faktoren ab. Manche Komponenten genießen nur ein kurzes Leben im All, andere können über viele Jahre in Betrieb sein. Spezielle Bauteile, die für den Militäreinsatz gebaut wurden, können im All sehr lange funktionieren. Kommerzielle Bauteile verhalten sich unterschiedlich. Obwohl nicht für den Einsatz im All hergestellt, können viele der Komponenten mehrere Jahre im All mit minimalem Schaden funktionieren.

Q *Mit welchen Strahlungseffekten hat man im All zu rechnen?*

A **Verschiebungen im Kristallgitter (Displacement damage):** Partikel (Neutronen, Protonen, Alphateilchen, schwere Ionen) und hochenergetische Gamma-Photonen verändern die Form der Kristallgitter und beeinflussen die Halbleitereigenschaften.

Ionisierende Strahlung (schnelle Veralterung): Ladungen werden induziert und akkumuliert. Diese Ladungen erhöhen die Verluste (Störströme) des Bauteils, bis das Bauteil selbst zerstört ist.

Single Event Effect: Hochionisierte Teilchen induzieren kurzzeitig starke Ladungen (Störimpulse) im Halbleiter und stören das Verhaltes des Bauteils, so dass Daten verfälscht werden.

Q *Welche Strahlungstests gibt es?*

A Es gibt in der Praxis hauptsächlich zwei Arten von Strahlungstests:

Total Ionizing Dose Test (TID): Dabei wird eine Cobalt-60-Tablette als Gamma-Quelle eingesetzt. Eine solche radioaktive Quelle ist aus der Medizin bekannt, aber hier viel schwächer. Bei der Bestrahlung werden mehrere Jahre Einsatz im All in kürzester Zeit simuliert. Die durch Strahlung induzierten Ladungen beeinflussen die Halbleitereigenschaften auf verschiedene Art und Weise.

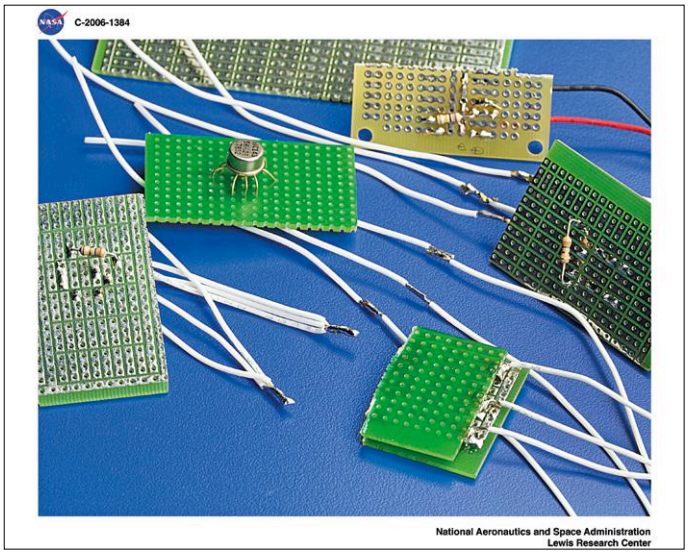


Bild 1. Astronauten der ISS üben die Reparatur elektronischer Bauteile (Foto: NASA).

Single Event Effect Test (SEE): In diesem Fall kommt ein Teilchenbeschleuniger zum Einsatz, der die Bausteile mit Protonen oder mit schweren Ionen beschießt. Das gravierendste Resultat ist die Verfälschung von Daten durch einen kurzen Störimpuls, der in die Schaltung induziert wird. Für digitale Schaltungen ist dieser Effekt sehr relevant.

Q *Gibt es Unterschiede in Bezug auf den Satellitenorbit?*

A Ja, je höher der Orbit, desto höher die Strahlungsdosis. Die Low-Earth-Orbit-Satelliten (LEO) werden der niedrigsten Strahlungsdosis ausgesetzt. Im Flugzeug erhalten Passagiere in 10 km Höhe auch eine höhere Strahlungsdosis, die Crews dürfen sogar nur eine bestimmte Zeit aktiv fliegen, um die eigene Gesundheit nicht zu gefährden. Die Raumfahrtstation ISS fliegt auf einer Höhe zwischen 350 km und 450 km. Dieser Orbit wurde ausgewählt, um die Astronauten einer möglichst geringen Strahlungsdosis auszusetzen.

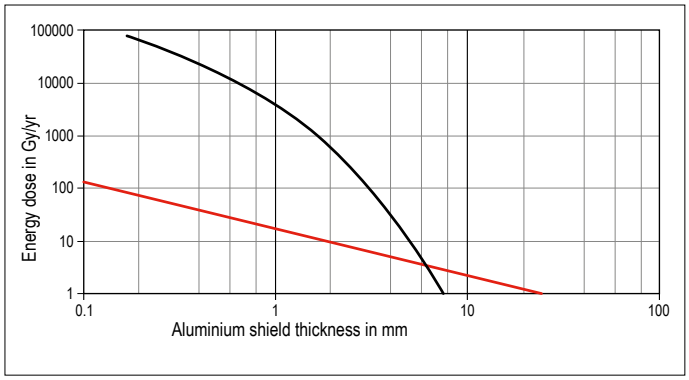


Bild 2. Die auf die Elektronik wirkende Energiedosis ist hier über der Dicke der Aluminiumabschirmung aufgetragen (schwarz: Elektronenstrahlung, rot: elektromagnetische Strahlung; Quelle: Wikipedia, gemeinfrei).

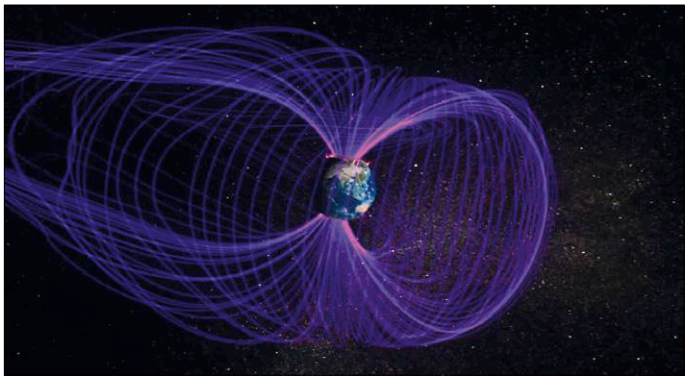


Bild 3. Die erste amerikanische Satellitenmission, Explorer 1, entdeckte 1958 einen Gürtel elektrisch geladener Teilchen im Umfeld der Erde (Zeichnung: NASA).

Ein weiterer Schutz stellt das Erdmagnetfeld dar. Solange ein Objekt sich im Erdmagnetfeld befindet, bleibt es vor hoch geladenen Teilchen geschützt. Die hoch geladenen Teilchen werden vom Magnetfeld abgefangen und zu den Polen umgeleitet. Telekommunikationssatelliten kreisen im geostationären Orbit (GEO). In dieser Umlaufbahn scheint der Satellit immer die gleiche Position in Bezug zur Erde einzunehmen. Der geostationäre Orbit liegt mit einer Höhe von fast 36.000 km deutlich außerhalb des Erdmagnetfelds. Dieser Orbit ist anspruchsvoller für die Elektronik, weil sie unter einer höheren Strahlenbelastung funktionieren muss. Deep-Space-Missionen sind noch heikler. Der größte Teil der Elektronik wird im Inneren von der Satellitenstruktur geschützt.

Q *Welche Art von Bauteilen wird in der Raumfahrt eingesetzt?*

A Je länger und höher der Flug ist, desto mehr Schutzmechanismen müssen eingesetzt werden. Der Entwurf muss die Anforderungen erfüllen, nicht weniger, aber auch nicht (viel) mehr. **ITAR-**Bauteile (International Traffic in Arms Regulations) können im LEO problemlos 100 Jahre funktionieren, doch wenn eine Mission beispielsweise nur fünf Jahre dauern soll, ist diese Art von teuren Bauteilen übertrieben. Daher werden mehr und mehr **COTS**-Bauteile (Commercial-Off-The-Shelf) verwendet. GEO-Satelliten müssen in einem deutlich raueren Umfeld 15 Jahre pausenlos funktionieren. Der Einsatz von ITAR-Bauteilen ist deshalb sinnvoll. Es kommt sogar immer wieder vor, dass die Elektronik solcher Satelliten nach deren geplanter Dienstzeit munter weiter funktioniert.
Es gibt auch Höhenforschungsraketen, die nur ein paar (maximal 15) Minuten fliegen. Auf spezielle Bauteile kann man verzichten, COTS-Komponenten sind völlig ausreichend.

Q *Welcher Schutzmechanismus ist wo am effektivsten/sinnvollsten?*

A Fast alle Satelliten werden mit **Redundanz** gebaut. Ausnahmen sind Pico- und kleine Nanosatelliten, wo es nicht genug Platz gibt, um die Elektronik zu duplizieren. Redundanz erlaubt es, die Betriebszeiten zu erhöhen. In der frühen Geschichte der Raumfahrt hatten alle Systeme mehrere redundante Module. Ein Bordcomputer zum Beispiel konnte bis zu achtfache Redundanz aufweisen. Vor allem für bemannte Missionen muss die Elektronik problemlos funktionieren. Redundanz erhöht zwar die Komplexität, den Energieverbrauch, das Gewicht und das Volumen des Raumfahrzeugs, bleibt aber eine der wichtigsten Methoden, um die Zuverlässigkeit eines Raumfahrtsystems zu erhöhen.
Toleranz: Es ist immer empfehlenswert, die Elektronik mit robusten Komponenten zu entwerfen. Man kann sogar die bekannte Degradierung der Bauteile im Entwurf berücksichtigen, sodass die Funktionalität der Elektronik immer in den Betriebsgrenzen bleibt. Auf einem Datenblatt eines kommerziellen Bauteils kann man das Verhalten des Bauteils unter Weltraumbedingungen aber nicht ablesen. Sogar die Hersteller kennen es nicht. Für den Einsatz im All sind diese Werte aber sehr wichtig, so dass man sie experimentell ermitteln muss.
Abschirmung: Es ist üblich, sensible Elektronik mit der Struktur des Satelliten zu schützen. Je dicker die Abschirmung, desto besser der Schutz, aber auch desto höher das Volumen der Struktur und noch schlimmer das Gewicht des Satelliten. Diese beiden Parameter lassen die Startkosten anwachsen. Man kann auch andere Materialien verwenden, aber die meisten sind schwerer als Aluminium. Spezielle Abschirmungen werden deshalb nur eingesetzt, wo es unbedingt erforderlich ist.

Q *Welche Bauteile sind besonders gefährdet, welche weniger, welche nicht?*

A Den wichtigsten Unterschied gibt es zwischen passiven und aktiven Komponenten. Bauteile wie Widerstände sind sehr robust, Halbleiter dagegen werden stark von Strahlung degradiert. Im allgemein ist die Bipolar-Technologie robuster gegen Strahlung als die CMOS-Technologie. Das hängt aber von vielen Faktoren ab wie der Schichtengeometrie, den Materialien, der Verkapselung und so weiter.
Sensibel gegen **ionisierende Strahlung** sind Dioden, Transistoren, alle ICs, Quarze, CCDs und optoelektronische Komponenten, während FET-Transistoren, CCDs und CMOS-APSs (Aktive Pixel Sensor) sowie andere optoelektronische Bauteile und alle ICs sind anfällig für **Single-Event-Effekte** sind.
Heutzutage fördern ESA und NASA die Nutzung von FPGAs, weil programmierbare Hardware aus der Ferne Reparaturen zulässt. Zudem packen FPGAs viele Funktionen in einen einzigen Baustein, sodass die Elektronik kompakter wird. Es gibt auch einen Prozessor der ESA mit dem Namen LEON, einen IP-Soft-Core, der auf einem FPGA implementiert werden kann [1].

Q *Wie können sich Beeinträchtigungen/Beschädigungen auswirken?*

A Die Elektronik kann bei der Degradierung zum Beispiel mehr Energie verbrauchen. Bei einem bestimmten Degradierungsgrad wird das Bauteil zerstört.
Wenn man das Verhalten des Bauteils im All kennt, kann man diese Eigenschaft im Entwurf berücksichtigen. Ein Beispiel: Wenn ein Verstärker einen Verstärkungsfaktor von 5 hat und am Ende der Mission (bekannt durch Simulation oder Qualifizierungstest) sich dieser Verstärkerfaktor auf 3 verringert, dann wird diese Information in den Schaltungsentwurf einfließen, sodass die Funktionalität der Schaltung die gesamte Einsatzzeit gegeben ist. So kann man der Degradierung des Bauteils entgegenwirken.
Wenn man Bauteile für Raumfahrtelektronik verwendet, deren Verhalten im All unbekannt ist, spielt man russisches Roulette. Der professionellste, aber auch aufwendigste Weg ist die Qualifizierung von einzelnen Bauteilen. Heutzutage gibt es Satellitenbetreiber, die gerne Zeit und Geld sparen wollen und sehr kurze Tests (sogar mit einer ganzen Platine „am Stück") durchführen. Wenn die Platine dann ausfällt, weiß niemand, welches Bauteil versagt hat. Die Robustheit der Platine wird immer vom schwächsten Glied bestimmt. Aber wenn man das Verhalten jedes einzelnen Bauteils kennt, hat man gute Chancen, die Platine robuster zu gestalten.

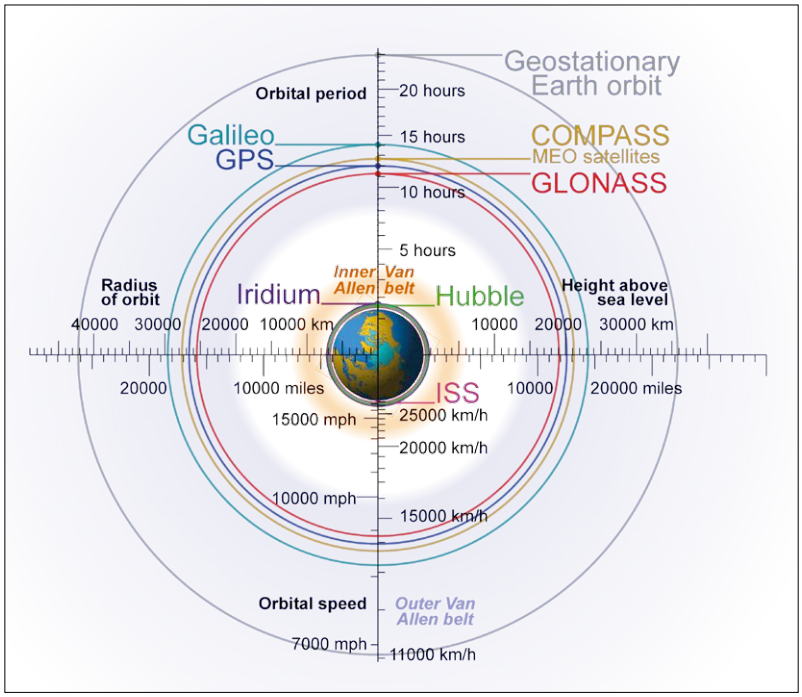


Bild 4. Vergleich der Erdumlaufbahnen verschiedener bekannter Satelliten (Zeichnung: Geo Swan, CC BY-SA 3.0).

Ein TID-Test dauert rund zwei Wochen und folgt ESA-Empfehlungen. Manchmal soll ein Test in nur sechs Stunden durchgeführt werden. Aber es ist wie beim Kuchenbacken: Wenn auf der Packung „200 °C für 30 min" steht, erhält man nicht das gleiche Ergebnis, wie wenn man den Kuchen für nur 10 min im Rohr lässt, dieses aber auf 600 °C erhitzt. Physikalische Vorgänge können nicht ausgetrickst werden! ⏮

(160299)

Literatur

„Electronics for Space Guide", Jaime Estela, in Kürze bei Elektor erhältlich.

Anzeige

BECKER MÜLLER

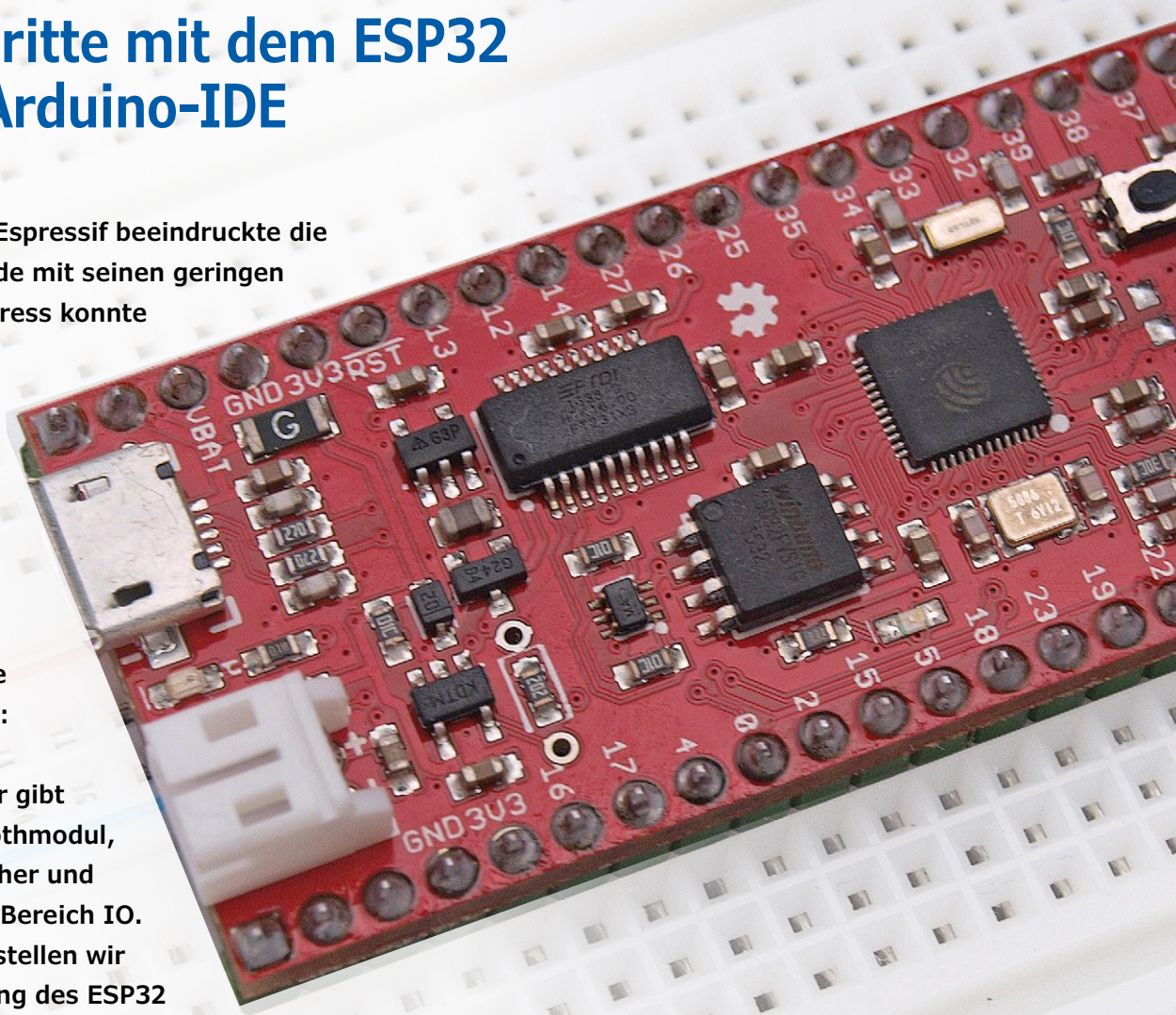
Leiterplatten Spezialisten

PUNKTGENAU
PLANGENAU
PREISGENAU

Becker & Müller
Schaltungsdruck GmbH
Tel.: +49 (0)7832 9180-0
www.becker-mueller.de

Der große Bruder des ESP8266

Erste Schritte mit dem ESP32 und der Arduino-IDE



Der ESP8266 von Espressif beeindruckte die Entwicklergemeinde mit seinen geringen Kosten: Bei AliExpress konnte man das Bauteil schon um zwei oder drei Euro erhalten. Mit dem großen Bruder ESP32 möchte Espressif seine WLAN-SoCs für die Zukunft fitmachen: Neben einem Zweikernprozessor gibt es nun ein Bluetoothmodul, mehr Arbeitsspeicher und Erweiterungen im Bereich IO. In diesem Artikel stellen wir die Programmierung des ESP32 mit der Arduino-IDE vor. Eine Besprechung des nativen Entwicklungstools IDF folgt im nächsten Heft.

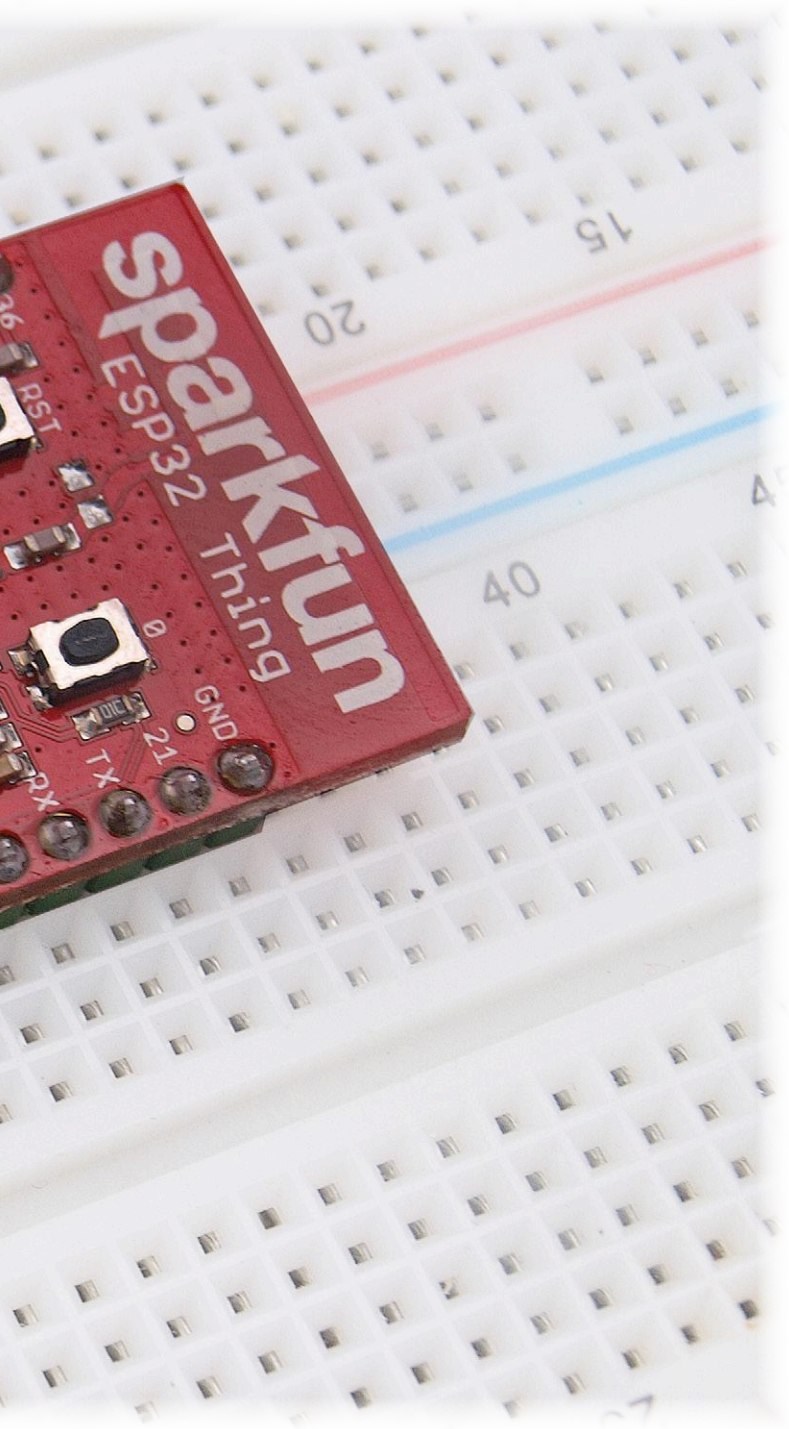
Von **Tam Hanna** (Slowakei)

Nach der Erstauslieferung des ESP32 kam es zu massiven Lieferproblemen bei den Breakout-Boards; Einzelchips waren davon nur wenig betroffen. Ob des sehr geringen Pin-Pitches ist das Entwickeln eigener Platinen für erste Gehversuche wenig empfehlenswert. Wir nutzen in diesem Artikel das SparkFun „ESP32 Thing“, das beispielsweise bei Mouser unter [1] bezogen werden kann.

SparkFun liefert keine Steckerleiste mit - die zum Anschluss an eine Steckplatine notwendigen Pins müssen separat gekauft werden. Neben dem ESP32 Thing gibt es mittlerweile auch Dutzende andere Boards verschiedener Hersteller. Espressif betreut unter [2] eine Liste geeigneter Kandidaten, die sich - beim Vorhandensein eines USB-Chips von FTDI - im Großen und Ganzen analog verhalten.

Workstation an ESP32

Auch das ESP32 Thing ist mit einem der bekannten USB-Seriell-Chips von FTDI ausgestattet. Dies ist für Nutzer insofern vorteilhaft, als Besitzer eines Arduinos oder eines ähnlichen Boards den Treiber mit hoher Wahrscheinlichkeit schon auf ihrem PC haben. In den folgenden Schritten nutzen wir als Betriebssystem Windows 8.1 Enterprise, die Arduino-IDE liegt in Version 1.8.2 vor.



Nervtötende Fehler

Beachten Sie, dass frühe Revisionen des ESP32 diverse Fehler aufweisen. Eine lesenswerte Liste findet sich unter [9].

Zum Zeitpunkt der Drucklegung dieses Artikels hatte Espressif das als Core bezeichnete SDK noch nicht in den Boardverwalter der Arduino-IDE eingepflegt. Zum Herunterladen des Codes ist das unter [3] bereitstehende grafische Frontend für Git erforderlich, das sich im Startmenü unter „Git GUI“ einnistet. Klicken Sie sodann auf *Clone Existing Repository*, und geben Sie im Feld *Source Location* „https://github.com/espressif/arduino-esp32.git“ ein.

Als übergeordneter Zielordner dient *C:\Users\TAMHAN\Documents\Arduino*. Achten Sie darauf, dass sie statt *TAMHAN* den Benutzernamen eingeben müssen, unter dem Sie die Arduino-IDE auf Ihrem Computer ausführen. Je nach Konfiguration finden sich in diesem Verzeichnis ein oder mehrere Sketches und eventuell sogar ein Unterordner *hardware*, in dem vom Benutzer manuell eingefügte SDKs zu liegen kommen; es ist allerdings kein Problem, wenn der Unterordner fehlt.

Nun können Sie herleiten, was Sie in das Feld *Target Directory* eingeben müssen - auf dem PC des Autors wäre das zum Beispiel *C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32*.

Klicken Sie sodann auf den *Clone*-Knopf, um das Herunterladen des Codes aus dem GitHub-Repository anzustoßen. Je nach Auslastung des Diensts kann es hier zu einer kürzeren oder zu einer längeren Wartezeit kommen. Wundern Sie sich nicht, dass das Git-Fenster stellenweise nicht reagiert; nach getaner Arbeit erscheint das in **Bild 1** gezeigte mehrfarbige Fenster, das sie kommentarlos schließen können.

Witzigerweise wird das SDK in zwei Blöcken ausgeliefert: Per GitHub kommen nur jene Informationen an, die spezifisch für die Arduino-Umgebung sind. Zum Herunterladen der allgemeinen ESP32-Komponenten dient das Programm *C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32\tools\get.exe*, das Sie idealerweise aus dem Windows-Explorer heraus per Doppelklick aufrufen. Das Programm öffnet ein Konsolenfenster, das Sie über den Download-Fortschritt informiert. Nach dem erfolgreichen Durchlaufen findet sich ein zusätzliches Verzeichnis namens *C:\Users\TAMHAN\Documents\Arduino\hardware\espressif\esp32\tools\xtensa-esp32-elf*, das die im nächsten Artikel besprochene Entwicklungsumgebung ESP-IDF enthält.

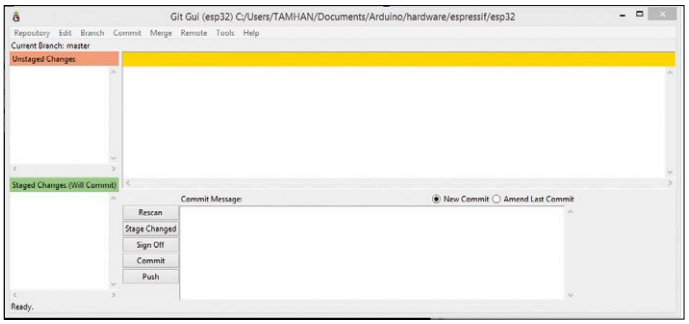


Bild 1. Das Arduino-SDK für den ESP32 wird über GitHub heruntergeladen.

Verbinden Sie das ESP32 Thing sodann mittels Micro-USB-Kabel mit dem PC, um den Treiber-Suchprozess anzustoßen. Wundern Sie sich nicht, dass der Assistent mitunter eine Maus anzeigt: der Treiber von FTDI ist aus für den Autor unverständlichen Gründen mit diesem Piktogramm ausgestattet. Nach der erfolgreichen Erkennung können Sie die Arduino-IDE starten, und unter *Werkzeuge* ➞ *Board* das *SparkFun ESP32 Thing* auswählen. Auf kleinen Bildschirmen kann es sein, dass Sie nach unten scrollen müssen; die Arduino-IDE zeigt hauseigene Platinen im Board-Auswahlmenü immer über den Boards von Drittherstellern an.

Die Arduino-IDE erkennt das ESP32 Thing nicht automatisch.

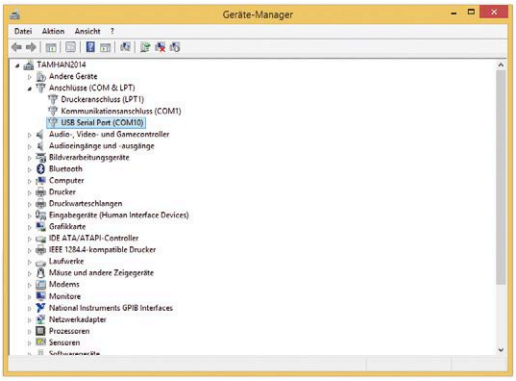


Bild 2. Die Nummer des COM-Ports lässt sich im Gerätemanager herausfinden.

Zur Ermittlung des korrekten COM-Ports wechseln Sie in den Gerätemanager, wo sich das Gerät wie in **Bild 2** gezeigt als *USB Serial Port* ausgibt. Geben Sie diesen Port sodann in der Arduino-IDE ein, um die Konfiguration abzuschließen. Die vorgegebene Upload-Geschwindigkeit von 921600 bps kann anfangs beibehalten werden: ESP32-erfahrene Entwickler reduzieren sie mitunter, wenn es beim Übertragen der (sehr großen) Binärdateien zu Trouble kommt. Dies führt allerdings zu einer längeren „Zykluszeit" bei der Neuprogrammierung, weshalb übermäßige Vorsicht nur wenig ratsam ist.

Hallo Welt!

Entwickler von Evaluationboards spendieren ihren Kindern normalerweise eine oder mehrere LEDs; in der Arduino-Welt hat sich hierfür der Pin Nummero 13 etabliert. SparkFun tanzt naturgemäß aus der Reihe: hier befindet sich die LED an Pin Nummero fünf.
Als erster Test bietet sich deshalb der Sketch in Listing 1 an, die Software zu diesem Artikel können Sie bei Elektor herunterladen [4].

Aus programmiertechnischer Sicht findet sich hier nichts Überraschendes: Wir schalten die LED regelmäßig ein und wieder aus. Nach dem zwischen 30 und 60 Sekunden dauernden Deployment-Prozess blinkt die blaue LED periodisch.

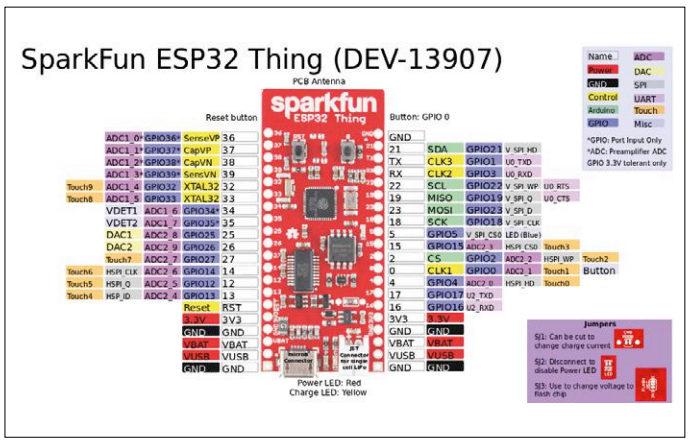


Bild 3. Das Pinout des ESP32 Things ist relativ komplex.

Wie schnell geht's los?

Als nächste Aufgabe wollen wir feststellen, wie viel Zeit vergeht, bis der ESP32 mit der Abarbeitung von Kommandos beginnt (dieser Wert ist in Zeiten von Echtzeitbetriebssystemen relevant: Der Arduino 101 genehmigt sich hier zum Beispiel bis zu acht Denksekunden, was in MSR-Applikationen schon mal zu gefährlichen Betriebszuständen führen kann).

Zur Lokalisierung der für die Messung interessanten Pins müssen wir uns im ersten Schritt mit dem Aufbau der Hardware auseinandersetzen. In der ESP-IDF findet sich die Struktur `gpio_num_t`, die für das Ansprechen der GPIOs zuständig ist. Der Code präsentiert sich – im Interesse besserer Lesbarkeit stark gekürzt - folgendermaßen:

```
typedef enum {
    GPIO_NUM_0 = 0,     /*!< GPIO0, input and output */
    GPIO_NUM_1 = 1,     /*!< GPIO1, input and output */
    . . .
    GPIO_NUM_39 = 39,   /*!< GPIO39, input mode only */
    GPIO_NUM_MAX = 40,
} gpio_num_t;
```

Aus dem „linearen" Aufbau der Struktur können wir ableiten, dass der ESP32 nicht mit GPIO-Pin-Bänken arbeitet: Alle Pins sind über ihre jeweilige ID eindeutig beschrieben. Eine nähere Betrachtung des Codes ist an dieser Stelle nicht notwendig - die folgende Methode zeigt, dass die angelieferten IDs direkt weiterverwendet werden:

```
extern void IRAM_ATTR __digitalWrite(uint8_t pin,
uint8_t val) {
    if(val) {
        if(pin < 32) {
            GPIO.out_w1ts = ((uint32_t)1 << pin);
        }
. . .
```

Mit diesem Wissen können wir uns dem Datenblatt des ESP32 Things zuwenden, das unter [5] im PDF-Format zum Download bereitsteht und **Bild 3** enthält. Nutzer einer anderen Entwicklungsplatine konsultieren stattdessen die Dokumentation des jeweiligen Herstellers.

Für uns sind hier die beiden Pins GPIO5 und VUSB interessant: an ersterem findet sich die von unserem Programm geschaltete LED, während der zweite direkt mit dem Micro-USB-Port verbunden ist und uns so Informationen über das Anliegen und/oder Nicht-Anliegen von Spannung liefert. Da der in den folgenden Schritten verwendete Oszillograph der Bauart LeCroy 9354AM vierkanalig ist, wollen wir zudem auch noch den Pin 3V3 überwachen, der mit dem Ausgang des auf der Platine befindlichen Spannungsreglers verbunden ist.

Vom Raspberry Pi umsteigende Entwickler müssen an dieser Stelle extrem vorsichtig sein: der ESP32 ist ein reiner 3,3-V-Controller, der beim Anliegen von Spannungen größer 3,6 V irreparabel beschädigt wird. Anders als der Prozessrechner der Bauart Upton ist der ESP32 in keiner Weise 5-V-tolerant; dies ist auch ein Problem, wenn das Board an Busse wie SPI oder I2C angeschlossen werden soll.

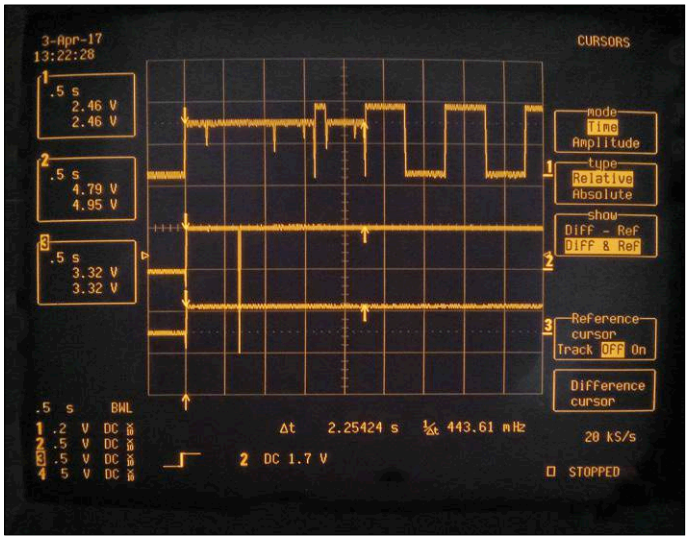


Bild 4. Bis das Blink-Programm startet, vergehen etwas mehr als 2 Sekunden.

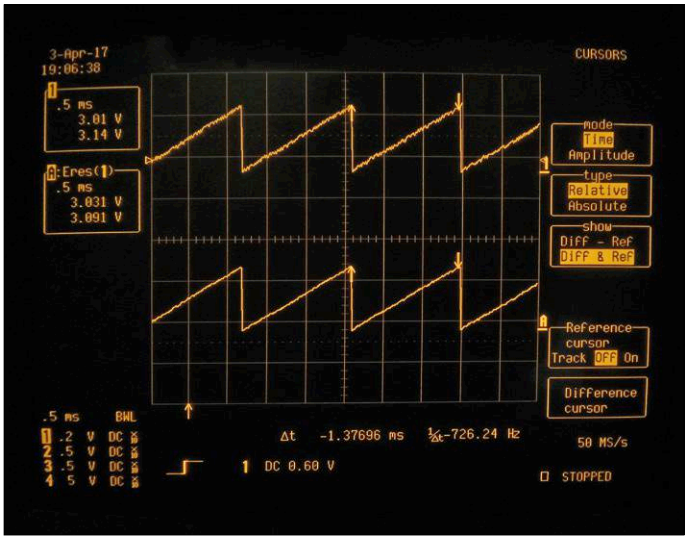


Bild 5. Mittels DAC können wir allerlei Signale ausgeben.

Auf der Haben-Seite steht derweil eine Versorgungsspannung von ab 2,2 V: Wer den ESP32 in eine eigene Platine einbaut, muss also nicht unbedingt 3,3 V zur Verfügung stellen. Der Spannungsbereich von 2,2..3,6 V erinnert übrigens verdächtig an Lithium-Ionen-Zellen, ein durchaus beabsichtigtes Anwendungsszenario.
Nach diesen Exkursen wollen wir jedoch zur Hauptaufgabe zurückkehren. Eine Analyse der Ausgabe mit dem Oszilloskop zeigt, dass der ESP32 etwa zweieinhalb Sekunden Startzeit braucht (siehe **Bild 4**).
Anzumerken ist in diesem Zusammenhang, dass das Evaluationboard einen relativ komplexen Bootloader verwendet. Während dessen Abarbeitung ist die LED halb hell: dies ist im ersten Kanal des Oszis erkennbar.

Mach's spezialisiert

Auch wenn es verlockend wäre, an dieser Stelle zum Modulation Domain Analysator zu greifen, um auch die Geschwindigkeit beim Umschalten des Ausgangs zu messen: der ESP32 dürfte - sowohl aufgrund der Nutzung von FreeRTOS, als auch aufgrund der doch hohen Kosten - nur sehr selten für harte Echtzeitaufgaben eingesetzt werden.

Viel interessanter ist es da, die im ESP32 reichlich vorhandenen Digital-Analog-Wandler einzusetzen. Unter Arduino-Nutzern hat sich hierfür die Methode `analogWrite` etabliert - leider wird sie zum Zeitpunkt der Drucklegung dieses Artikels vom Espressif SDK noch nicht unterstützt. Die Hersteller empfehlen stattdessen drei Alternativen: erstens den als LEDC bezeichneten PWM-Prozessor, der auf bis zu 16 Pins gleichzeitig arbeiten kann [6]. Zweitens das Sigma-Delta-Modul [7], das bis zu acht Kanäle bespielt. Drittens den klassischen DAC [8], der allerdings nur bis zu zwei Kanäle bearbeitet.

Wir wollen hier auf die dritte Methode setzen: Das Realisieren von kleinen Funktionsgeneratoren ist eine seit jeher lustige Aufgabe. Wer stattdessen eines der anderen Elemente verwenden möchte, kann den angegebenen Links folgen - sie führen zur Headerdatei, die das jeweilige Modul beschreibt.

Arbeit mit dem DAC

Wie immer ist es empfehlenswert, zuerst einmal die Implementierung der API anzusehen. Öffnen Sie die weiter oben abgedruckten Links in einem Browser ihrer Wahl, tauschen dabei aber die Endung .h durch .c aus.
Im Fall des DACs lautet die relevante Methode `__dacWrite`, deren Korpus folgendermaßen aussieht:

```
void IRAM_ATTR __dacWrite(uint8_t pin, uint8_t value) {
    if(pin < 25 || pin > 26){
        return;//not dac pin
    }
    pinMode(pin, ANALOG);
    uint8_t channel = pin - 25;
```

An dieser Stelle sind zwei Fakten interessant. Erstens hat sich in der ESP32-Welt eingebürgert, die beiden Kanäle mit den Indizes der zu ihnen gehörenden GPIO-Pins zu identifizieren.

Listing 1. Unser erstes ESP32-Programm!

```
//File ElektorESO32Scratch1

int ledPin = 5;

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
}
```

Die Methode erwartet deshalb den Wert 25 oder 26, der im ersten Schritt in eine Kanal-ID umgewandelt wird.

Interessant ist, dass Espressif dem Nutzer der Methode offensichtlich nicht zutraut, den Zustand effizient zu verwalten. Aus diesem Grund wird die Methode `pinMode` in jedem Aufruf abermals aktiviert - dass dies der Effizienz nicht sonderlich zuträglich ist, ist logisch.

Damit können wir zur Realisierung der Wellenformausgabe zurückkehren. Unser DAC ist achtbittig, weshalb die Laufvariable von 0 bis 255 marschieren muss:

```
void loop() {
    for(int i=0;i<255;i++) {
      dacWrite(25, i);
    }
}
```

Verbinden Sie sodann einen Oszillographen mit dem auserwählten Pin, um sich an der in der **Bild 5** gezeigten Wellenformausgabe zu erfreuen. Die „Duplizierung" der Spur liegt in einer Besonderheit des LeCroy begründet – neben der „Urtrace" wird in Bild 4 auch eine um drei Bit verbesserte Version angezeigt.

WLAN-Verbindung

Das von Espressif bereitgestellte SDK wird mit einer Gruppe von Programmbeispielen ausgeliefert, die die Nutzung der Netzwerk-APIs demonstrieren: Sie finden sich unter *Datei → Beispiele → Beispiele für SparkFun ESP32 Thing*.

Wir wollen hier die Kommunikation mit dem Internet ausprobieren; da es dabei einige Stolperfallen gibt, realisieren wir das Programm in zwei Stufen. Unsere erste Demo soll nur beweisen, dass wir uns erfolgreich mit einem WLAN verbinden konnten. Zuerst ist der Name des WLANs als String erforderlich: Da Groß-und Kleinschreibung eine Rolle spielt, sollten Sie ihn nach Möglichkeit über die Zwischenablage in das Programm einfügen. Um den Namen Ihres WLANs angezeigt zu bekommen, empfiehlt sich unter Windows die Nutzung des *netsh*-Kommandos - wundern Sie sich nicht, wenn nur ein Teil der Ausgabe in das Terminalfenster passt. Der relevante Block sieht beim Autor jedenfalls folgendermaßen aus:

```
C:\Users\TAMHAN>netsh wlan show all

. . .

=============== SCHNITTSTELLEN ANZEIGEN ================
Es ist 1 Schnittstelle auf dem System vorhanden:

. . .

    SSID                   : Tamoggemon Holding k.s. BAa
```

Im Programmcode müssen wir den WLAN-Header `<WiFi.h>` inkludieren; dann legen wir zwei weitere Variablen an, die den Namen und das Passwort des zu verwendenden Netzwerks enthalten:

```
#include <WiFi.h>
const char* ssid     = "Tamoggemon Holding k.s. BAa";
const char* password = "XXX";
```

Im nächsten Schritt erledigen wir Housekeeping-Tasks; neben dem Einrichten der seriellen Verbindung zum PC fügen wir ein Delay ein, um den ESP32 etwas Zeit zur Abarbeitung der Konfigurationsbefehle zu geben:

```
void setup() {
    Serial.begin(115200);
    delay(10);
```

Nun starten wir den eigentlichen Verbindungsaufbau. Hierbei ist relevant, dass `WiFi.begin(…)` sofort retourniert. Den erfolgreichen Verbindungaufbau können wir aber über `WiFi.status()` mittels Polling abfragen:

```
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
```

Zu guter Letzt geben wir die IP-Adresse aus, die dem Router von der WLAN-Infrastruktur zugewiesen wurde:

```
    Serial.println("");
    Serial.println("connected - IP is:");
    Serial.println(WiFi.localIP());
}
```

Der Inhalt der Methode `loop` kann momentan leer bleiben. Führen Sie das Programm sodann bei geöffnetem seriellen Monitor aus, um die in **Bild 6** gezeigte Ausgabe zu erhalten. Beachten Sie, dass wir hier 115200 bps verwenden: Nach dem Start weist der serielle Monitor eine Baudrate von 9600 auf und kann die eingehenden Informationen so natürlich nicht dekodieren.

Daten aus dem Netz

Nun können wir uns an das Herunterladen von Informationen aus dem Netz wagen. Für die Kommunikation mit TCP-Servern wird das `WiFiClient`-Objekt verwendet.

`WiFiClient` interessiert sich nicht für Applikationsebenen-Protokolle. Aus diesem Grund ist zum Aufbau einer Verbindung nur der Name des anzusprechenden Hosts und die Nummer des zu verwendenden Ports erforderlich; im Fall einer fehlgeschlagenen Verbindung wird `false` zurückgegeben:

```
void loop() {
    WiFiClient client;
    const int httpPort = 80;
    if (!client.connect("tamoggemon.com", httpPort)) {
        Serial.println("connection failed");
        return;
    }
```

Das Fehlen eines HTTP-Treibers belastet uns insofern, als wir eine gültige Anfrage von Hand formulieren müssen. Da es in diesem Artikel nicht um das HTTP-Protokoll als Ganzes geht,

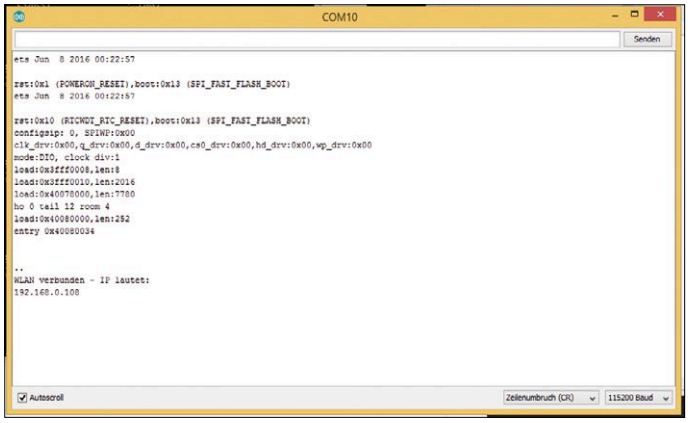


Bild 6. Hurra, wir sind im Netz!

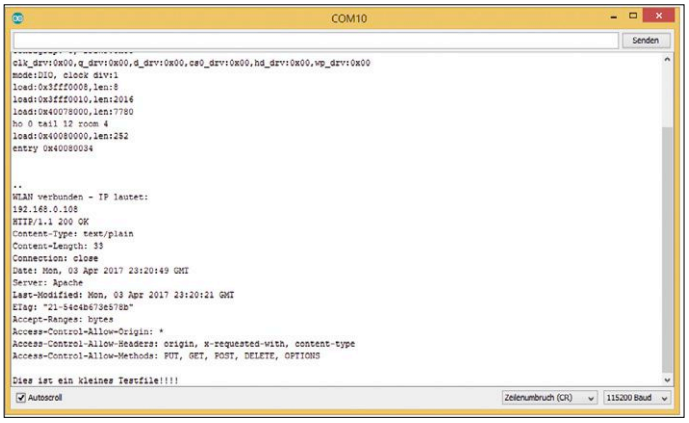


Bild 7. HTTP ist sehr gesprächig.

sollten Sie sich über den im Listing gezeigten String keine weiteren Gedanken machen:

```
    String url="/test/2017/Elektorfile.txt";
    client.print(String("GET ") + url + " HTTP/1.1\r\n"
+
                 "Host: " + "tamoggemon.com" + "\r\n" +
                 "Connection: close\r\n\r\n");
    unsigned long timeout = millis();
```

Die Bearbeitung von HTTP-Anfragen kann je nach Belastung des Servers mehr oder weniger Zeit in Anspruch nehmen. Wir geben unserem Server hier 5 Sekunden, und lösen einen Timeout aus, wenn innerhalb dieser Zeitspanne nicht mindestens ein Byte eingeht:

```
    while (client.available() == 0) {
        if (millis() - timeout > 5000) {
            client.stop();
            return;
        }
    }
```

Nach dem Auftauchen von Daten lesen wir diese peu à peu aus dem Client-Objekt, und leiten sie in Richtung des seriellen Monitors weiter:

```
    while(client.available()) {
        String line = client.readStringUntil('\r');
        Serial.print(line);
    }
```

Da permanentes „Polling“ des Servers lästig ist, rufen wir die Methode `delay(…)` auf. Sie sorgt dafür, dass der ESP32 fünfzig Sekunden lang ruht:

```
    delay(50000);
}
```

Damit kann der Sketch in Richtung des ESP32 wandern. Wundern Sie sich nicht darüber, dass die Konsole mit diversen Ausgaben befüllt wird: Jede HTTP-Abfrage liefert neben den eigentlich angeforderten Informationen einen (immens langen) Header mit (siehe **Bild 7**).

Fazit

Ein Raspberry Pi Zero W oder ein OrangePi Zero sind ernstzunehmende Konkurrenten für den ESP32: bei den Breakout-Boards gibt es wenig Preisunterschied. Wer eine Kleinserie plant, kann mit dem ESP32 allerdings viel Geld sparen: Ab einigen Stück lohnt sich der Mehraufwand für eine eigene Platine, da die einzelnen ESP32-Chips wesentlich preiswerter als die Breakout-Boards sind.
Unsere Reise durch die Welt des ESP32 ist an dieser Stelle noch nicht zu Ende. Im nächsten Heft nutzen wir die native API: sie bietet Zugriff auf einige weitere Funktionen, und ermöglicht zudem das Erreichen einer höheren Performance. ⏮

(160454)

Weblinks

[1] http://eu.mouser.com/new/sparkfun/sparkfun-things/

[2] http://esp32.net/#Hardware

[3] https://git-scm.com/download/win

[4] www.elektormagazine.de/160454

[5] https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP32ThingV1.pdf

[6] https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-ledc.h

[7] https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-sigmadelta.h

[8] https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-dac.h

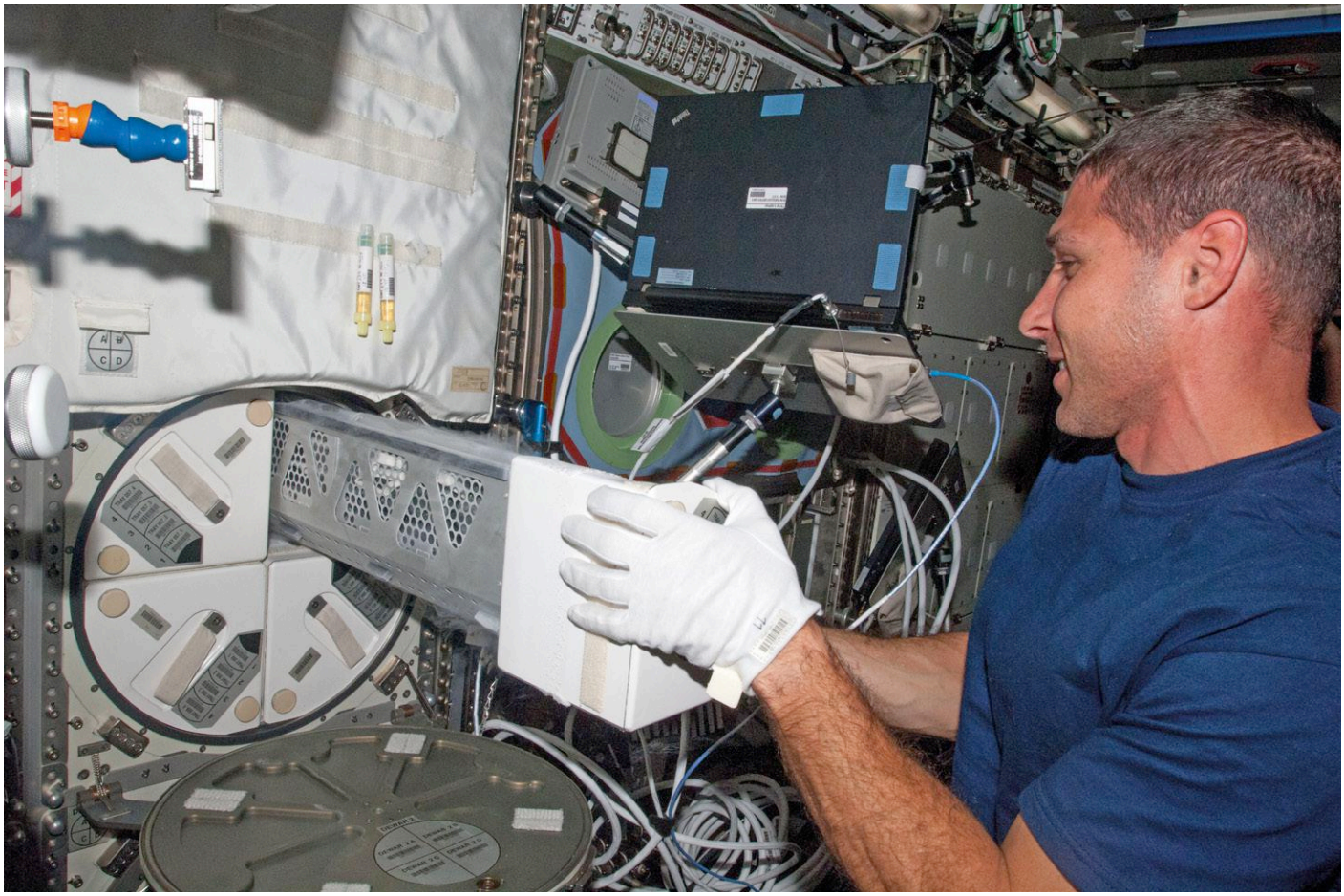
[9] http://espressif.com/sites/default/files/documentation/eco_and_workarounds_for_bugs_in_esp32_en.pdf

Elektronik im All

Weltraumforschung für jedermann

Von **Jaime Estela**

Satelliten werden aus robusten Materialien gebaut, da sie unter extremen Umweltbedingungen fehlerfrei funktionieren müssen. Für die Elektronik werden eigens hergestellte und deswegen extrem teure Bauteile eingesetzt. Bei Kleinsatelliten sind kommerzielle Komponenten „aus dem Regal“ eine preisgünstige Alternative, deren Verhalten aber vor dem Einsatz in Simulationsverfahren für Raumfahrtanwendungen und nach internationalen Standards qualifiziert werden muss.



NASA-Astronaut Mike Hopkins holt ein Dewar-Gefäß aus dem Eisschrank, um biologische Proben einzusetzen (Foto: NASA).

Im All gibt es verschiedene physikalische Phänomene, die das Verhalten von elektronischen Bauteilen und Materialien beeinflussen. Für eine Satellitenmission ist es notwendig, diese Umgebung sehr gut zu kennen, um Schäden und Störungen in der Elektronik zu vermeiden beziehungsweise zu reduzieren. Das genaue Verhalten der Bauteile unter solch widrigen Bedingungen hängt von der Umlaufbahn des Satelliten ab und kann mit Software-Tools simuliert werden. Die Simulationsergebnisse helfen den Satellitenentwicklern, das System entsprechend zu entwerfen.

Bild 1 zeigt die physikalischen Phänomene, die die Lebensdauer und die Zuverlässigkeit elektronischer Bauteile beeinflussen:

- **Atomischer Sauerstoff**: UV-Licht bricht die O_2-Moleküle in einzelne Sauerstoffatome auf. Der atomare Sauerstoff ist sehr reaktiv und ero-

diert aus diesem Grund die Oberfläche der Satellitenstruktur. Dieser Rost wirkt sich auf das thermische Verhalten der Struktur und des Satelliten aus. Dies ist ein sehr wichtiges Thema, da das Thermalsystem des Satelliten beeinträchtigt wird.

- **Plasma**: Die ionisierten Gase erzeugen elektrostatische Ladungen und belasten die Oberfläche des Satelliten elektrisch. Die Entladung solcher Lasten kann den Betrieb des Satelliten und der Instrumente stören.

- **Strahlung**: Eine Vielzahl von Effekten gehört zu diesem Phänomen. Gamma-Strahlen degradieren die elektronischen Komponenten. Protonen und schwere Ionen können im besten Fall digitale Daten verfälschen oder im schlechtesten buchstäblich die Elektronik des Satelliten zerstören.

- **Mikrometeorite und Weltraummüll**: Die gefährlichsten Ereignisse in All stellen kleine künstliche oder natürliche Körper dar. Der Einfluss von Mikrometeoriten oder Weltraummüll ist nicht zu unterschätzen, er kann Satelliten beschädigen oder zerstören. So etwas ist bereits geschehen, mit der Folge des Verlusts des Raumfahrzeugs.

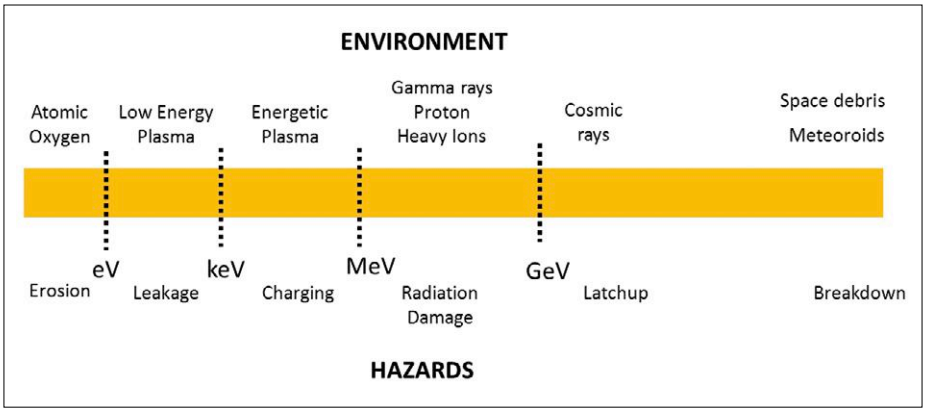


Bild 1. Umweltbedingungen im All und ihre möglichen Schadensrisiken (Quelle: Spectrum Aerospace)

Degradierung der Elektronik

Zu Beginn der Raumfahrtgeschichte gab es keine spezielle, für die Raumfahrt entwickelte Elektronik. Elektronische Bauteile für militärische Zwecke wurden für den Einsatz im All aufgerüstet. In diesem Up-screening-Prozess wurden verschiedene ergänzende Tests durchgeführt und die besten Bauteile ausgewählt.
1973 wurden für die Hardware der Raumstation Skylab ebenfalls militärische Komponenten verwendet. Beim Qualifikationstest musste die Hardware aufgrund der auftretenden Störungen mehrfach verbessert werden. Diese Verbesserungen hatten weitere Investitionen von über 3 Millionen Dollar zur Folge, weil neue elektronische Komponenten beschafft werden mussten und so zusätzliche Qualifizierungskampagnen notwendig wurden.

Die Elektronik der ersten Space-Shuttle-Mission (1981) verwendete wiederum militärische Bauteile. Um die Zuverlässigkeit des Systems zu erhöhen, wurden die meisten sechsfach redundant verbaut. Unter Verwendung eines statistischen Verfahrens (Wahlschema) wurden die gültigen Daten überprüft. Die Einbeziehung redundanter Systeme bedeutete aber auch eine Steigerung von Gewicht und Stromverbrauch und weitere Herausforderungen im Hardware- und Software-Design.

Da die Militär-Elektronik sich nicht wirklich für Raumfahrtanwendungen eignete und das Up-Screening auch keine Verbesserung der Bauteile mit sich brachte, wurde in den 1960er Jahren mit der systematischen Entwicklung vom Raumfahrtelektronik begonnen, die qualitativ hochwertige Herstellungsverfahren erfordert. In den USA wurden als Basis für die neue Komponentengeneration militärische Bauteile ausgewählt und komplementäre Tests für die Qualifizierung definiert. Diese Strategie ermöglichte eine Senkung der Produktionskosten bei Raumfahrtmissionen, dennoch wurde gleichzeitig eine hohe Robustheit der Komponenten erzielt. Da aber die Nachfrage nach solchen Bauteilen naturgemäß sehr gering war, bleib der Preis hoch.

Heutzutage stellt die Qualifizierung von nur einem Bauteil nach ESA- oder NASA-Standards eine Investition von einer Million Dollar oder mehr dar. Die Zertifizierungszeit liegt bei rund zwei Jahren [1]. Natürlich verläuft je nach Bauteiltyp der ganze Test komplexer oder einfacher. Der Test beispielsweise einer Diode ist weniger kostenintensiv und langwierig als der Test eines Mikrocontrollers.

Qualifizierungstests

Die gewonnene Erfahrung in vielen Weltraummissionen und mit verschiedenen Technologien der letzten sechzig Jahre ermöglichten Institutionen wie der ESA und der NASA, Qualifizierungsrichtlinien und Standards zu entwerfen. Diese Richtlinien unterstützen die professionelle Qualifizierung elektronischer Bauteile für Raumfahrtmissionen. Die von der ESA festgelegten Standards findet man auf dem Portal für Komponenten der Europäischen Raumfahrtagentur ESCIES (European Space Components Information Exchange System) [2]. Eine Datenbank mit qualifizierten elektronischen ICs einschließlich der Testberichte ist auf der ESCIES-Website verfügbar [3].

Für einen Qualifizierungstest werden die zu testenden Bauteile (Device Under Test, DUT) vorbereitet und die Messumgebung konfiguriert. Zuerst werden die genauen Stückzahlen der Bauteile bestimmt und abhängig von der Art des Tests einige wenige oder die ganze Menge der Bauteile genommen. Das bedeutet, dass manchmal wenige Bauteile für die Qualifikation ausreichen (beispielsweise bei Strahlungstests), während zum Beispiel bei einem Ausgasungstest alle Bauteile getestet werden müssen. Während der Tests werden die DUTs charakterisiert.
Eine wichtige Anforderung für die Qualifizierung der Bauteile ist, dass alle zu qualifizierenden Bauteile identisch sein (Geometrie, Größe, Materialien, und so weiter) und auch zur gleichen Herstellungscharge gehören müssen.

Die Testergebnisse werden in einem Testbericht zusammengefasst. Nach der Qualifizierung beschreibt ein Testbericht das

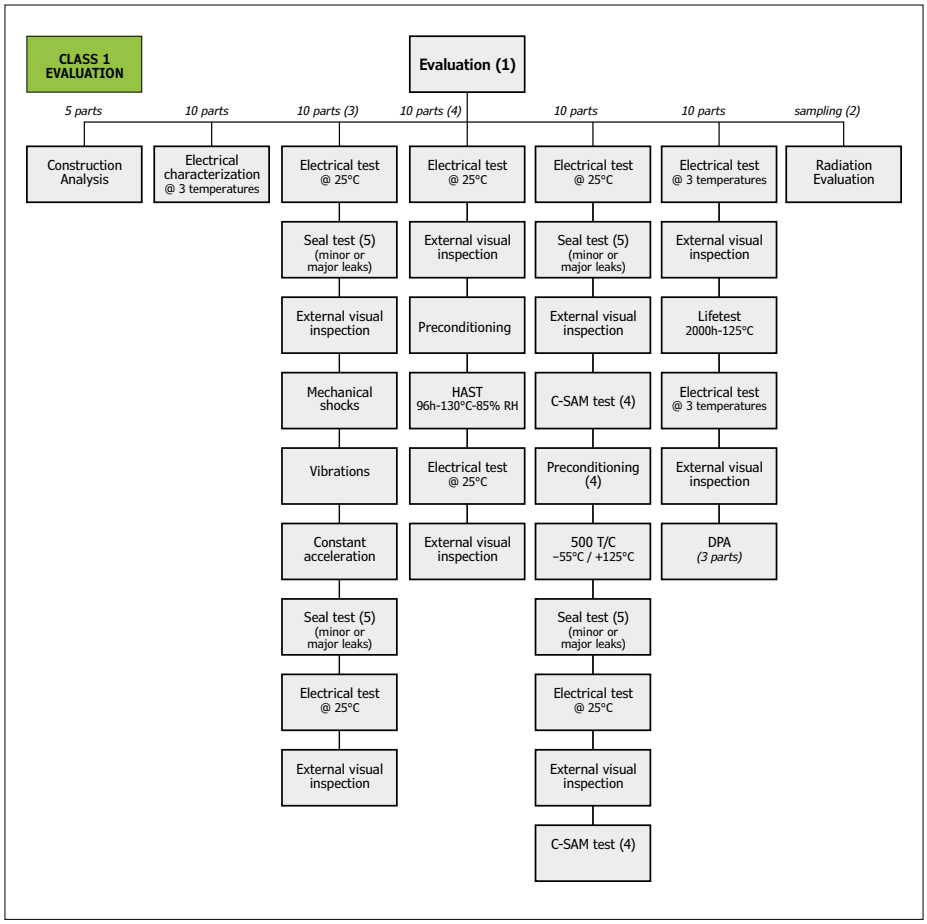


Bild 2. Testabläufe der Klasse 1 (ECSS-Q-ST-60-13C Evaluation tests flow Class 1, Quelle: ESA).

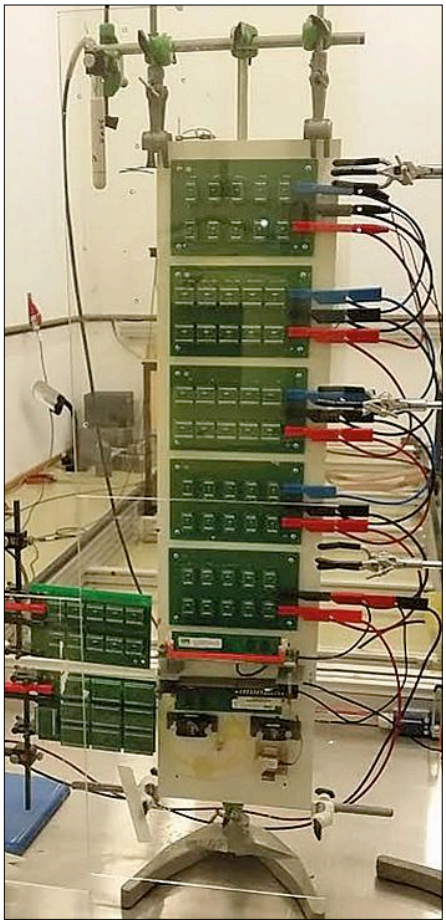


Bild 3. Der Ständer mit den DUTs ist bereit für den Strahlungstest (Quelle: Spectrum Aerospace).

Verhalten der relevanten Bauteileparameter unter Weltraumbedingungen. Eine vollständige, sehr aufwendige Qualifikation ist bekannt als „Screening Test". Ein solcher Prozess dauert ein paar Jahre und erfordert eine hohe finanzielle Investition. Im Screeningtest werden folgende Einzeltests durchgeführt:

- Elektrischer Test
- Dichtheitsprüfung
- Visuelle Inspektion
- Mechanischer Schock
- Vibrationstest
- Konstante Beschleunigung
- Thermischer Test
- Strahlungstest
- Ausgasung
- Stresstest
- Thermischer Schock
- Lötbarkeit

Im Dokument „ECSS-Q-ST-60-13C_ Space Produktsicherung" beschreibt die ESA verschiedene Qualitätsklassen [4]. Der Unterschied zwischen den Klassen hängt von der Tiefe des Qualifizierungsprozesses ab. Die Klasse 1 bezieht sich auf eine vollständige Qualifikation (**Bild 2**), während die Klasse 3 nur einen kurzen Qualifizierungsprozess (hauptsächlich Strahlungstests) erfordert.

Charakterisierung

Wie solche Qualifizierungsprozesse ablaufen, lässt sich aus einer kurzen Beschreibung des TID-Tests (Total Ionizing Dose) eines driftfreien Operationsverstärkers LTC2052 ersehen. Zunächst wird ein Gestell vorbereitet, das Platinen mit den zu untersuchenden Bauteilen enthält (**Bild 3**). Dabei sind die beiden in **Bild 4** abgebildeten Konfigurationen (Bias Condition) wichtig, ein „Off mode", bei dem alle Pins mit Masse verbunden sind und ein „On mode", bei dem das Gerät in einem bestimmten Arbeitspunkt konfiguriert ist, das Bauteil jedoch kein Signal verarbeitet (also eine Art Stand-by).

Bild 5 zeigt den Testaufbau. Als Strahlungsquelle kommt Cobalt-60 zum Einsatz. Die beiden Testkonfigurationen werden einer Bestrahlung ausgesetzt. Während der Bestrahlung werden die Prüflinge von Zeit zu Zeit mit einem Prüfgerät (Automatic Test Equipment, ATE) vermessen. Diese Charakterisierungsergebnisse zeigen die Degradierung der Bauteile bei einer immer höheren Strahlendosis. Die Verminderung der Leistungsfähigkeit hängt von der Technologie des Bauteils und der akkumulierten Strahlungsdosis ab. Zum Beispiel kann man bei zunehmender Strahlungsdosis eine immer höhere Stromaufnahme beobachten (**Bild 6**). Am Ende des Tests werden Veränderungen einiger elektrischer Parameter festgestellt, in einigen Fällen wurden die Komponenten sogar komplett zerstört.

Kleinsatellitenmarkt

Der Markt für kleine Satelliten wächst ziemlich rasant. Neue Produkte und neue Dienstleistungen werden gesucht, um dem wachsenden Bedarf entgegenzukommen. Für die Herstellung von Kleinsatelliten-Hardware sind militärische ITAR-Komponenten (International Traffic in Arms Regulations) nicht die richtige

Fragen an und Antworten von Jaime Estela

- *Es war ein Wettbewerb angedacht, bei dem man einen freien Platinenplatz in einem StarLab-Flug gewinnen kann. Was ist daraus geworden?*

Ja, das ist aktuell. Es gibt aber Verzögerungen, weil eine Firma, die den Wettbewerb mitorganisieren wollte und Erfahrung mit solchen Veranstaltungen hat, ausgestiegen ist. Wir sind im Gespräch mit anderen Firmen, aber das dauert sicherlich noch ein bisschen.

- *Wann findet der erste Flug statt?*

Der erste Flug sollte in Dezember stattfinden, aber solange es noch Unklarheiten gibt, haben wir keinen festen Termin. Im Prinzip finden die Flüge im 3-Monats-Rhythmus statt.

- *In welcher Größenordnung werden die Kosten liegen?*

Der Preis hängt von der Größe der Platine ab. Für eine kleine Platine (5x3 cm²) kann man mit ungefähr 3000 € aufwärts rechnen.

- *Wie werden die Experimente beobachtet?*

Elektronik im All wird über Telemetriedaten überwacht. Sensoren messen bestimmte Parameter und die Messergebnisse werden zur Erde übertragen. So weiß man immer genau, wie die Elektronik im All arbeitet. Ein Entwickler muss sich aber auf sein eigenes Experiment konzentrieren, daher wird ein StarLab-Entwicklungskit zur Verfügung gestellt. Diese Hilfselektronik unterstützt den Elektronikentwickler, damit er schnell und unkompliziert sein Experiment mit bereitgestellter Telemetrieelektronik vorbereiten kann.

- *Kann man nur Bauteile oder auch Baugruppen testen?*

Man kann beides machen. Einzelne Bauteile zu testen nimmt mehr Zeit in Anspruch und die Kosten sind höher, die Ergebnisse sind aber sehr genau. Für ganze Platinen sind die Kosten und die Testzeit geringer, aber die Ungenauigkeit ist groß. Das Problem bei ganzen Baugruppen beziehungsweise ganzen Platinen ist, dass man im Falle eines Ausfalls den Fehler eventuell nicht genau lokalisieren kann.
Nicht alle Komponenten sind gleich robust. Daher sollte man zuerst die Robustheit der Komponenten kennen, bevor man ein System für den Einsatz im All entwirft. Wenn man es umgekehrt macht, kann es sein, dass ein einziges Bauteil die Robustheit der ganzen Baugruppe stark verschlechtert.
Ich habe ein neues Konzept entworfen, das zwischen Komponenten- und Platinen-Level liegt. Dabei werden ganze Platinen genauer qualifiziert.

- *An wen kann ich mich wenden, wenn ich Fragen oder auch ein konkretes Interesse habe?*

Jaime Estela
jaime.estela@gmail.com
jaime.estela@spectrum-aerospace.com
https://www.linkedin.com/in/jaime-estela-9045b63b/
Skype: jaimeestela

Lösung, sondern qualifizierte COTS-Bauteile (Commercial Off-The-Shelf, seriengefertigte Produkte).
Die von der Firma Northern Sky Research durchgeführten Marktuntersuchungen deuten darauf hin, dass in den nächsten zehn Jahren mehr als 2.500 kleine Satelliten (von bis zu 100 kg) ins All gestartet werden [5], eine weitere Marktstudie der Firma Spacework geht von 3.000 kleinen Satelliten bis zu 50 kg in den Jahren 2014 bis 2020 aus [6]. In beiden Prognosen wurden zukünftige Satellitenkonstellationen (in „Massenproduktion“ gefertigte und deshalb verhältnismäßig billige Satellitengruppen wie zum Beispiel GPS) berücksichtigt. Eine zunehmende Serienfertigung von Satelliten bedeutet wieder einen höheren Bedarf an qualifizierter Elektronik.

Space-COTS

Für NewSpace [7], eine neue Bewegung vieler (US-amerikanischer) Startups, die sich zum Ziel gesetzt hat, Raumfahrt preiswerter und verfügbarer zu gestalten, sind Space-COTS der Zugang zu kommerziellen Komponenten, die für den Einsatz im All qualifiziert sind. Der Begriff „Space-COTS“ und sein Konzept entstanden aus der Arbeit mit

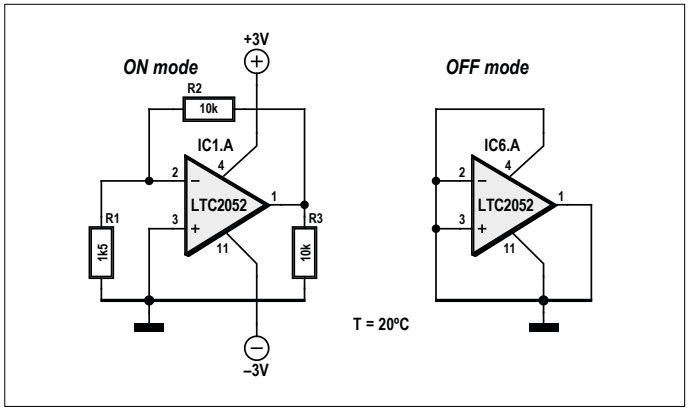


Bild 4. Zwei Konfigurationen des LTC2052 im TID-Test (Quelle: Spectrum Aerospace).

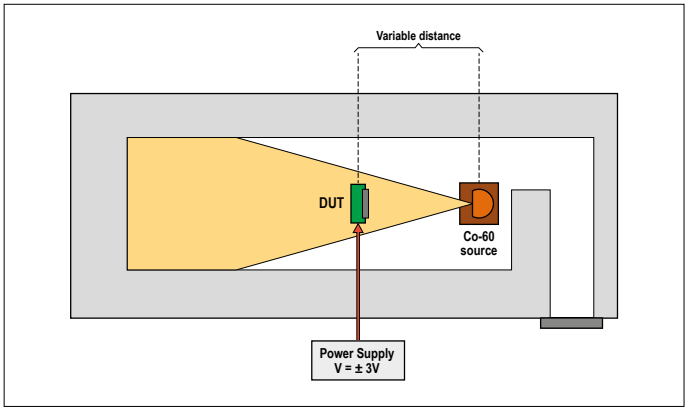


Bild 5. Testaufbau mit einer Cobalt-60-Strahlenquelle (Quelle: Spectrum Aerospace).

kommerzieller Elektronik in Kleinsatellitenmissionen und auch dem langjährigen Umgang mit ESA-Standards für die Qualifizierung elektronischer Bauteile. Space-COTS stellen einen Kompromiss zwischen überhaupt nicht qualifizierten Bauteilen (zum Beispiel in CubeSats) und voll qualifizierten ITAR-Bauteilen dar. Aus der Erfahrung mit Kleinsatellitenmissionen ist bekannt, dass viele kommerzielle Bauteile im All durchaus mehrere Jahre arbeiten können, obwohl sie nicht für Raumfahrtanwendungen konzipiert wurden.
Space-COTS werden in verschiedenen Stufen getestet und qualifiziert, je nach Qualifikationstiefe werden folgende Klassen definiert:

- Class A: Full-screening-Test
- Class B: TID- und SEE-Test (SEE = Single Event Effect)
- Class C: TID-Test

Diese Klassen folgen den ESA- und NASA-Vorgaben und auch den neuen ISO-Standards (International Organization for Standardization).

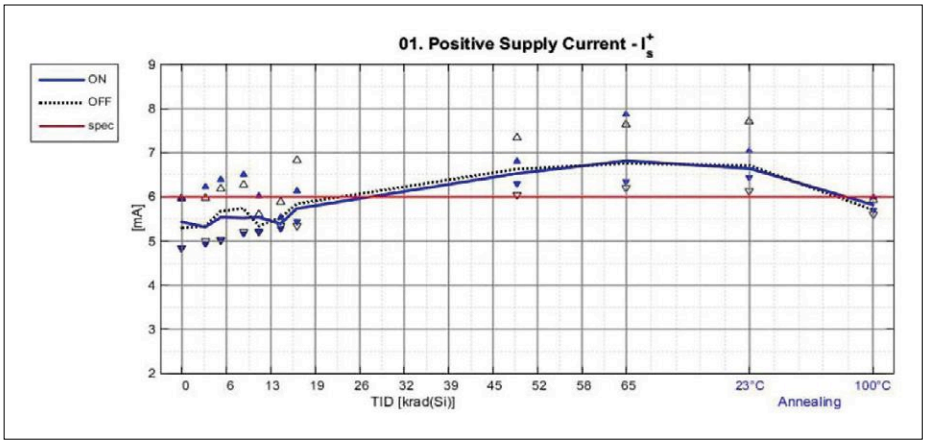


Bild 6. Ein Testergebnis: Die Stromaufnahme erhöht sich mit steigender Strahlendosis (Quelle: Spectrum Aerospace).

StarLab

Viele Firmen entwickeln neue Produkte für die Raumfahrt. Diese Produkte können aber keine Flugerfahrung (Flight Heritage) aufweisen. Um ein Produkt im All zu testen, muss man mindestens 250.000 € (Nanosatellit plus Flug ohne Testablauf) investieren. Außerdem strapaziert eine Wartezeit auf den Flug von mindestens 18 Monaten Geduld und Geldbeutel. Es gibt zwar Satelliten für Technologieerprobung, aber diese Flüge finden sehr selten statt und sind nur sehr wenigen privilegierten Firmen vorbehalten [8].

Inzwischen aber gibt es Plattformen, um die Raumstation ISS für ein breiteres Publikum nutzbar zu machen. Auf diesen Plattformen können kleine private Nutzlasten installiert werden. Eines dieser Projekte ist das StarLab-Projekt. Ein Container enthält diverse Experimente; Strom- und Datenanschluss ist für jedes Experiment verfügbar und die Testdaten werden fast in Echtzeit zur Erde übertragen. Das StarLab-Projekt mit seiner innovativen Qualifizierung erlaubt den regelmäßigen Zugang ins All für jedermann und dies zu moderaten Preisen.
Gute Elektronik kann durchaus auch in der Maker-Community entstehen. Solche Entwicklungen könnten auch für die Raumfahrt interessant sein, doch ist das Thema Raumfahrtelektronik in Maker-Kreisen noch weitgehend unbekannt. Das StarLab-Projekt verleiht dieser neuen Elektronik die Flügel, um Tests im All zu absolvieren. Eigene Entwicklungen im All zu testen ist zwar immer noch eine große Herausforderung, aber mit dem StarLab-Projekt wird diese aufwändige und kostspielige Aufgabe einfach und günstig. Den Entwicklern eröffnet sich ein völlig neues Betätigungsfeld und der Raumfahrtsektor kann von innovativen, qualifizierten Produkten profitieren. Neugierig geworden? Lust auf einen Weltraumspaziergang? Der Textkasten gibt Antwort auf Ihre brennendsten Fragen zum StarLab-Projekt. Das Tor ins All steht Ihnen weit offen! ⏮

(160036)

Quellen und Weblinks

[1] The history of space quality EEE parts in the United States, www.cti-us.com/pdf/HistoryEEESpacePartsinUSA.pdf, Leon Hamiter, Components Technology Institute

[2] European Space Components Information Exchange System, https://escies.org/webdocument/showArticle?id=167

[3] ESCIES Radiation List, https://escies.org/labreport/radiationList

[4] ECSS-Q-ST-60-13C_Space product assurance, ESA Requirements and Standards Division ESTEC, P.O. Box 299, 2200 AG Noordwijk, The Netherlands.

[5] Nano- And Microsatellite Markets Report, Northern Sky Research, LLC, One Mifflin Place, Suite 400, Cambridge, MA 02138.

[6] Small Satellite Market Observations 2015 - Forecast, Dr. John Bradford, Spaceworks Enterprises Inc., 1040 Crown Pointe Parkway, Suite 950, Atlanta, GA 30338 USA.

[7] NewSpace - Wikipedia, https://en.wikipedia.org/wiki/NewSpace

[8] DLR IOV/IOD Workshop, Marc Jochemich, DLR Space Administration, Brussels, September 2016.

Wo bitte ist das Menü?

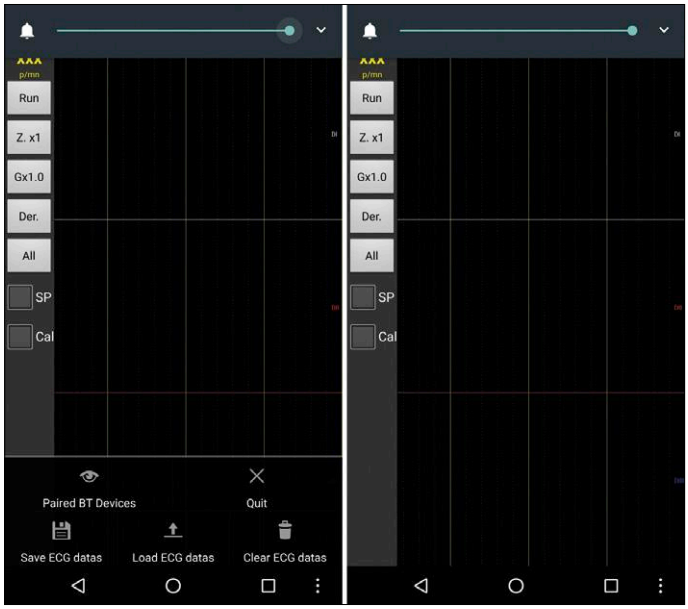


Bild 1. Die App des Android EKGs, mit und ohne Bedientaster.

Bild 2. Unterschiedliche Gestaltung des Menü-Tasters bei Android.

Von **Luc Lemmens** (Elektor-Labor)

Gelegentlich passiert es, dass Software streikt, wenn sie in die Jahre gekommen ist. Das kann vorkommen, nachdem das Betriebssystem oder die Hardware auf den Stand der Zeit gebracht wurden. Im Elektor-Labor waren wir nicht zum ersten Mal mit diesem Phänomen konfrontiert. Diesmal deuteten die Zeichen der Zeit darauf hin, dass die App unseres „Elektor Android EKG" vom Juli/August 2013 ein Opfer des „Fortschritts" geworden sein könnte.
Ein Elektor-Leser hatte sich erst in jüngster Zeit diesem Projekt zugewandt. Nachdem er Zeit und Mühe investiert hatte, vermisste er die Bedienelemente der zugehörigen App, die normalerweise, wie **Bild 1** zeigt, unten auf dem Touchscreen erscheinen. Ohne die Bedienelemente war der Zugang zu den Funktionen versperrt. Beim mobilen Gerät unseres Lesers fehlte der Menü-Taster, mit dem die Bedienelemente hervorgeholt werden können.

Android-Geräte, Smartphones ebenso wie Tablets, sind seit jeher mit den drei Tastern *Home*, *Back* und *Menü* auf dem Display unten ausgestattet, bei manchen Varianten ist außerdem ein Taster *Search* vorhanden. Bei Geräten der ersten Generation waren dies mechanische Drucktaster, später wurde der mechanische Taster *Home* von zwei kapazitiven Tastern flankiert. Leider konnten sich die Gerätehersteller nicht auf eine modellübergreifende Gestaltung einigen. **Bild 2** zeigt einige Beispiele für die Menü-Taster auf Smartphones.
Bei einigen Android-Geräten aus neuerer Zeit musste der Menü-Taster seinen Platz für den *Overview*- oder *App-Switch*-Taster räumen, mit dem der Benutzer zwischen den aktiven Apps umschalten kann. Wahrscheinlich besaß unser Leser, der das Android EKG gebaut hatte, ein solches modernes Smartphone. Die Lösung des Problems: Die Taster erscheinen auf dem Bildschirm der Kardioskop-App, wenn der Taster *Overview* etwas länger als normal betätigt wird. Dann wechselt die Funktion dieses Tasters von *App Switch* nach *Menü*.

Die Anfrage unseres Lesers war schnell beantwortet. Doch sie stimmte auch nachdenklich darüber, wie enorm unterschiedlich die Bedienung von Android-Geräten gestaltet sein kann. Oder ist es chaotischer Wildwuchs? Ich möchte es fast so nennen! Ganz genau erinnere ich mich, dass ich vor etwa einem Jahr damit beschäftigt war, mich mit der Bedienung meines damals brandneuen Smartphones anzufreunden. Sogar meine Kollegen habe ich in Mitleidenschaft gezogen, wenn ich in deutliche Missfallensbekundungen ausbrach, weil sich das Produkt modernster Kommunikationstechnik wieder einmal nicht so verhielt, wie ich es erwartet hatte... ⏮

(160349)gd

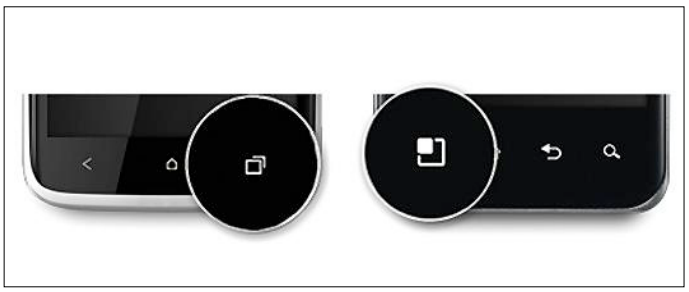
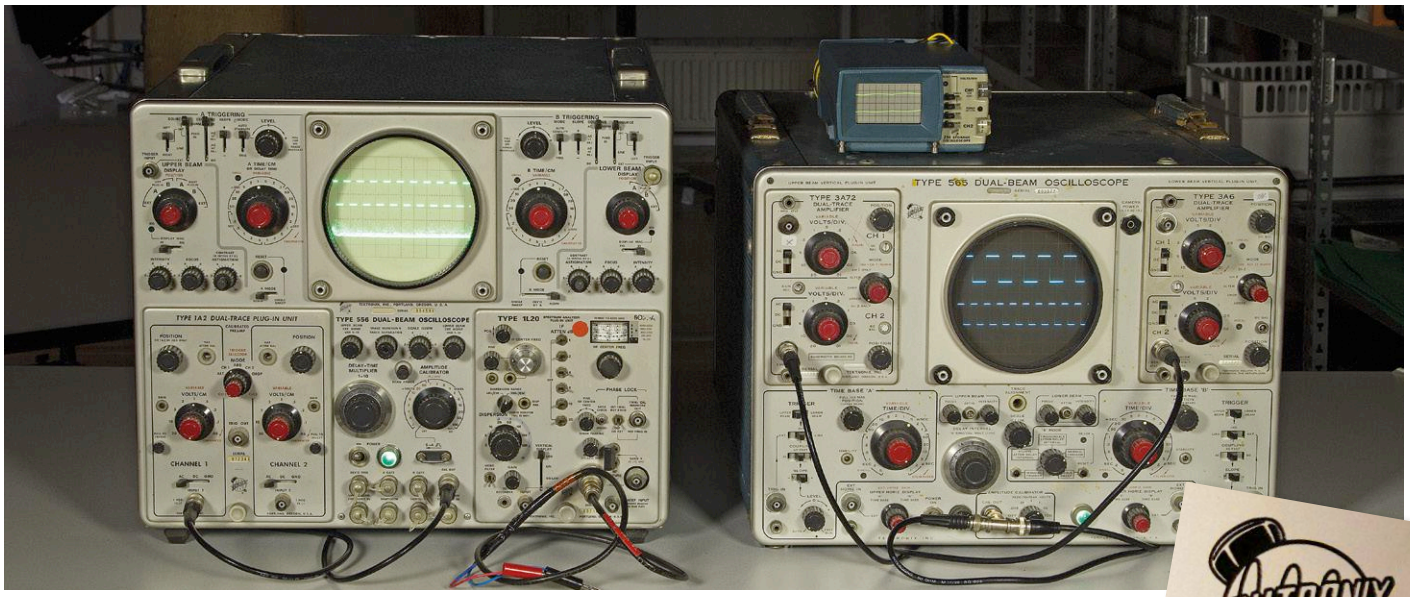


Bild 3. Umschalttaster für Apps.

Die Zweistrahl-Oszilloskope Tektronix 556 und 565

Groß, laut und klotzig…



„Du meine Güte! Schau Dir das an! Und dann auch noch mit *zwei* Kanälen protzen — wie witzig! Schon mein altes Scope hätte da nur einen Smiley für übrig!“ Solche Sprüche hört man oft, wenn Leute das erste Mal vor einem der Röhren-Oszilloskope 556 oder 565 von Tektronix stehen. Kein Wunder, denn es handelt sich um zwei der größten, schwersten und teuersten Messinstrumente, die je in einem Elektronik-Labor ihre Dienste anboten – oder es zumindest heizten ;-). Diese Monster stellten den Höhepunkt der Röhren-Oszilloskop-Technik des damaligen Marktführers Tektronix dar, auch wenn sie schon mehr als eine Handvoll Halbleiter enthielten. Und schließlich gab es noch ein schwereres Modell.

Von **Jan Buiting** (Elektor)

Zwei Strahlen gefällig?

Viele Röhren-Oszilloskope wurden von ihren stolzen Besitzern undifferenziert als 1/2/4/x-Kanal- oder -Strahl-Oszilloskope bezeichnet. Meiner Schätzung nach waren aber 98 % aller je verkauften Oszilloskope mit CRT (Cathode Ray Tube = Kathodenstrahlröhre) definitiv Einstrahl-Apparate, deren Strahl intern so schnell zwischen verschiedenen Eingangssignalen umgeschaltet wurde, dass das träge menschliche Auge zwei oder mehr unterschiedliche kontinuierliche Kurven sah. Das klappt für alltägliche Signale im kleinen Labor bis zu vier Kanälen ganz gut, wenn die CRT schnell und gut genug ist. Mit den gängigen CRTs ergaben sich keine wirklichen Probleme, da die Röhrenentwicklung damals schon ein sehr hohes Niveau erreicht hatte.

Für die damalige Zeit war es noch akzeptabel, dass das schnelle vertikale Umschalten des Elektronenstrahls zu Flackern und anderen Artefakten bei bestimmten Signalen führte, besonders wenn sich die Signale in der Frequenz oder gar Wiederholrate deutlich unterschieden. Damals war „digitale Fotografie“ noch pure Science-Fiction, und selbst ein „Screenshot“ mit Hilfe einer der in den 1960ern neuen Kassetten-Kameras eine Herausforderung.

Forschungseinrichtungen mit größerem Budget (wie die NASA) griffen deshalb lieber zum echten Zweistrahler mit „nur“ 15 MHz als zum Pseudo-Mehrkanaler, der angeblich ja so viel schneller und vor allem billiger war. Doch wenn man tatsächlich zwei Signale ohne Tricks auf

Bild 1. Zentrale Frontplatte des Modells 556 mit Knöpfen und Eingängen für beide Kanäle.

Bild 2. Einschub 1L20. Ein Spektrum-Analyzer mit 10 MHz bis 4 GHz für die 500er-Serie von Tektronix – sehr selten.

einer Röhre anzeigen wollte, dann blieben halt nur richtige Zweistrahlgeräte. Doch die waren eben teuer, schon aufgrund der komplexeren CRT mit doppelter Ablenkung. Aber auch die doppelte Vertikalverstärkung, also zwei getrennte HV-Einheiten (seltener sogar mit getrennter Zeitbasis), schlug zu Buche.

Man muss den Unterschied wirklich sehen, um es zu glauben. Das Bild eines Zweistrahl-Oszilloskops ist sowas von ruhig und es gibt keine Merkwürdigkeiten, wenn man die Kurven ganz oder teilweise übereinanderlegt. Zudem war auch hier der Chopper- (schnelles vertikales Umschalten zwischen langsamen Kurven) oder Alternativ-Betrieb (Kurven werden abwechselnd dargestellt bei schnellen Signalen) möglich, und dann hatte man 2 x 2 oder gar 2 x 4 Kanäle zur Verfügung. Außerdem konnte man z.B. Netzstörungen bei 50 Hz auf dem oberen Strahl und seine Auswirkungen auf HF-Stufen im unteren Strahl im MHz-Bereich beobachten.

Tektronix stellte einige wenige Oszilloskope der Zweistrahlklasse her: Die Typen 502, 551, 555 und 565 der berühmten 500er-Serie, das Modell 7844 der Premium-7000er-Serie und schließlich den Typ 5113 für Normalsterbliche.

Ich kenne zumindest einen weiteren Hersteller von Zweistrahl-Oszilloskopen: Philips hatte dereinst die transistorisierten Modelle PM3230, PM3231, PM3232 und PM3233 sowie die Röhrenversion PM3236 im Programm. Wenn Sie noch weitere Hersteller kennen, lassen Sie es mich wissen. Bislang ist mir auch noch nie ein Dreistrahl- oder gar Vierstrahlgerät über den Weg gelaufen.

Ganz speziell: Ein Tek 556

Auch wenn Mart, der Fotograf unter den Elektor-Mitarbeitern, schon an die außergewöhnlichen Apparate für Retronik gewöhnt ist, so musste er doch schmunzeln, als ich ihn für diese Ausgabe um Bilder der „nur“ rund 45 kg „leichten“ Schätzchen bat, denn die hätten seinen teuren Fototisch beschädigen können, der doch eher für Arduino, RPi & Co. ausgelegt war. Glücklicherweise stand noch ein stabilerer Schreibtisch mit 160 x 80 cm zur Verfügung, der mit Rollen nachgerüstet war. Dieser war in der Lage, die beiden Monster des Typs 556 und 565 nebst Kabeln, Zubehör und einem kleinen 214 zu transportieren. Also bugsierte ich den so beladenen Tisch zu ihm.

Das **Hauptbild** zeigt die zwei Goliaths und mit dem 214 einen Mini-David in Aktion. Gesamtenergiebedarf: 1.600 W aus einer Netzsteckdose in unserem früheren Fotostudio. Das Modell 556 (links) zeigt das 1-kHz-Kalibrierungssignal oben und unten das HF-Spektrum unseres 2,4-GHz-WLANs, empfangen via zwei Bananenbuchsen.

Auch wenn das 556 ein vollsymmetrisches Zweikanalgerät mit vielen doppelt ausgelegten Einheiten ist, sind doch viele Knöpfe und Schalter für beide Kanäle zuständig. Diese sind daher in der Mitte angeordnet (**Bild 1**). Das sehr gut erhaltene Gerät hat die Seriennummer 004584 und entstammt der Fabrik in Portland, Oregon.

Es handelt sich um ein konfigurierbares Oszilloskop mit zwei Einschüben für viele Zwecke. Zufällig ist hier links der horizontale 50-MHz-Zweistrahlverstärker des Typs 1A2 und rechts ein 10...4.200-MHz-Spektrum-Analyzer des

Bild 3. Modell 556 mit geöffnetem Deckel. Sichtbar ist die enorme Röhre mit zwei echten Elektronenstrahlen samt Ablenkungen. Dass moderne Oszilloskope LCDs oder gar LED-Bildschirme haben, ist kein schlechter Fortschritt.

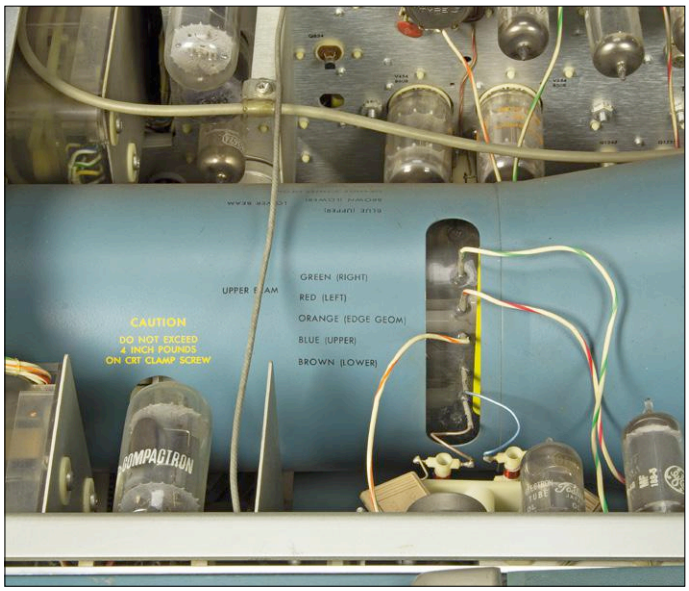


Bild 4. CRT-Anschlüsse im Modell 556. Einfache Litzen führen an die Ablenkungs-Pins im Glas.

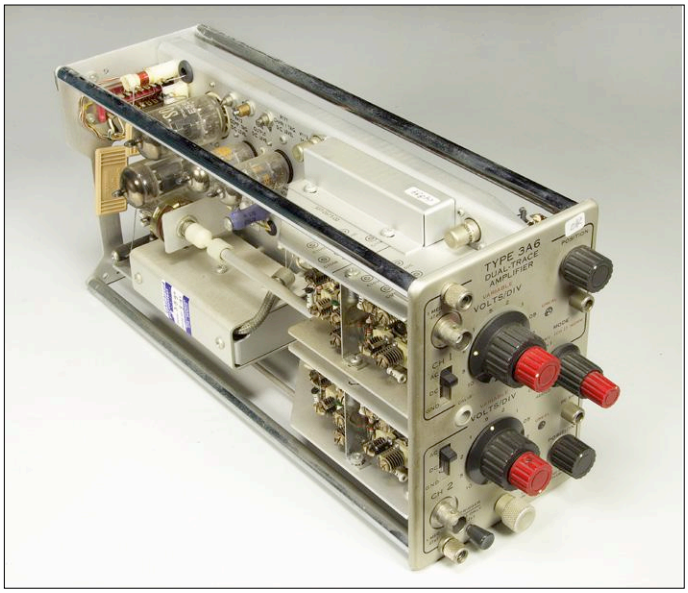


Bild 5. Zweikanal-Einschub 3A6. Zwei davon in einem 556 erlauben vier sauber dargestellte Kurven mit zwei Elektronenstrahlen.

Typs 1L20 zu sehen, der in **Bild 2** noch einmal getrennt abgebildet ist. Die Aufnahme der unteren Kurve war schwierig, da sie aufgrund der Aktivitäten von PCs und Smartphones ziemlich nichtperiodisch war.

Um die Größenverhältnisse auch im Druck der Zeitschrift deutlich zu machen, habe ich für **Bild 3** das Gehäuse geöffnet und auf die CRT-Abschirmung eine 9-V-Batterie gelegt. Es hätten auch 20 Stück Platz gehabt. **Bild 4** erlaubt einen näheren Blick auf die CRT – speziell auf die fünf Litzen der Ablenkung des oberen Strahls. Genauso sieht es gegenüber für den unteren Strahl aus. Der Aufdruck der anderen Seite ist teilweise sichtbar, sowie auch Vertikal-Kompensationsspulen und die auffällige Compactron-Röhre!
Es war keine Instandsetzung notwendig, nur etwas Reinigung. Ich erhielt das Oszilloskop mit den beiden abgebildeten Einschüben. Später kam ich noch an die Einschübe 1L10 und 1L30, wodurch ich 1 MHz bis 10,4 GHz abdecken konnte.

Zweistrahl-Oszilloskope der Tektronix-Serie 500

Typ	Gewicht (ohne Einschübe)	Eigenschaften	Elektor Retronik-Sammlung
502(A)	23,6 kg	Bis 500 kHz, hochempfindlich; preiswert; kompakt; 1 x Sweep; ohne Lüfter.	√
551	Anzeigeeinheit: 23,6 kg Netzteil: 19,2kg	Bis 30 MHz, hochempfindlich; 2 x V-Einschub; 1 x Sweep; kein Delay; getrennte Netzteile; 1 Lüfter.	√
555 „Triple Nickel“	Anzeigeeinheit: 31 kg Netzteil: 20,5 kg	Bis 30 MHz, hochempfindlich; 2 x V-Einschub; 2 x H-Einschub; Delay; getrennte Netzteile; 2 Lüfter.	X
556	40 kg	Bis 50 MHz; 2 x V-Einschub (Serien „letter“ und 1xx); 2 x Sweep; integriertes Netzteil; 1 Lüfter.	√
565	31 kg	Bis 15 MHz; 2 x 560er-Serie (Typ 2xx/3xx) Einschübe; 2 x Sweep, Delay; 1 Lüfter.	√

565 mit Schwierigkeiten

Verglichen mit dem 556 ist das Modell 565 minimal kleiner und hat mehr Trigger-Optionen sowie getrennte, geteilte und tauschbare Zeitbasen. Dafür hat das 565 eine Bandbreite von 15 MHz statt den 50 MHz des 556, was zur Frage führt, warum Tektronix dieses Modell überhaupt entwickelt hat. Die Antwort ist, dass damit die populäre 560er-Reihe wohl um ein Zweistrahlgerät erweitert wurde. Für die 560er-Subserie gibt es spezielle Einschübe zur direkten Ansteuerung der H- und V-Ablenkung der CRT. **Bild 5** zeigt den Zweikanalverstärker 3A6 meines 565. Verglichen mit Einschüben der 500er-Serie wie dem 1L20 (Bild 2 und Hauptfoto) fallen diese schmaler aus. Hinten sieht man zwei dicke Röhren des Typs E55L „SQ“ (Special Quality) zur Ansteuerung der V-Ablenkung mit bis zu 10 MHz und niedriger Kapazität für geringe Verluste und Verzerrungen. Es gab sehr viele 560-kompatible Einschübe mit vier oder gar acht Kanälen oder für GHz-Signale – damals eine beeindruckende Leistung der Entwickler von Tektronix.

Die beiden Geräte waren von verschiedenen Vorbesitzern und wurden von mir zusammen mit einigen Einschüben in

meinem Suzuki SX4 nach Hause transportiert. Bevor es mir gestattet war, sie auch nur in die Garage zu bringen, musste ich eine kurze Reinigung vornehmen, da sie im Laufe der jahrzehntelangen Lagerung zusammen mit anderem Elektroschrott doch ordentlich verschmutzt waren.

Mein 565 wurde mit einem Regeltrafo sanft angefahren. Nach gut vier Stunden war ich bei 185 V der nominellen 230 V angelangt und netterweise zeigte sich ein Strahl. Gerade als ich die schöne blaue Linie mehr ins Zentrum verschieben wollte, vernahm ich einen kleinen Knall und der Strahl war weg. Zwar habe ich sofort ausgeschaltet, doch schon erfüllte ein widerlicher Geruch den Raum und nach und nach das ganze Haus. Ich fürchtete die Verabschiedung eines Elkos irgendwo im Gerät, aber eine visuelle Inspektion brachte keine Hinweise. Es war auch keine Sicherung durchgebrannt und auch nach erneutem Einschalten glühten alle Röhren friedlich vor sich hin. Selbst die fünf geregelten Spannungen stimmten mit einer Toleranz von 2 %. Nach dem Schnüffeln im Gehäuse entfernte ich die Abdeckung der HV-Abteilung und erkannte den Bösewicht: ein schwarzer Kondensator mit Explosionsmerkmalen. Gemessen an der Kleinheit des Bauteils und dem winzigen Loch darin war der Gestank beeindruckend! Der Kondensator gehörte zum zentralen Oszillator der Hochspannungserzeugung. Zehn Minuten später war ein 0,1-µF-Ersatz mit 630 V eingelötet (**Bild 6**). Nachdem die Einsteller auf Standardposition gesetzt und das Scope eingeschaltet war, zeigten sich zwei scharfe Kurven auf dem Schirm. Auch der schwarze Sprague-Kondensator mit 1 nF und 1 kV unter meinem Zeigefinger wäre ein Kandidat fürs Auswechseln gewesen, aber für den Moment war er noch okay.

Die beiden Kurven waren mit ihren 7,5 cm gegenüber normalen 10 cm deutlich zu kurz. Dies wurde mit der Einstellung von SWEEP WIDTH und den zugehörigen Potis für X1 GAIN im Gerät behoben, sollte aber noch überprüft und mit einem Zeitmarkengenerator neu kalibriert werden. Die Einsteller hinterließen einen staubigen Eindruck und sind ein Fall für eine Behandlung mit Isopropylalkohol.

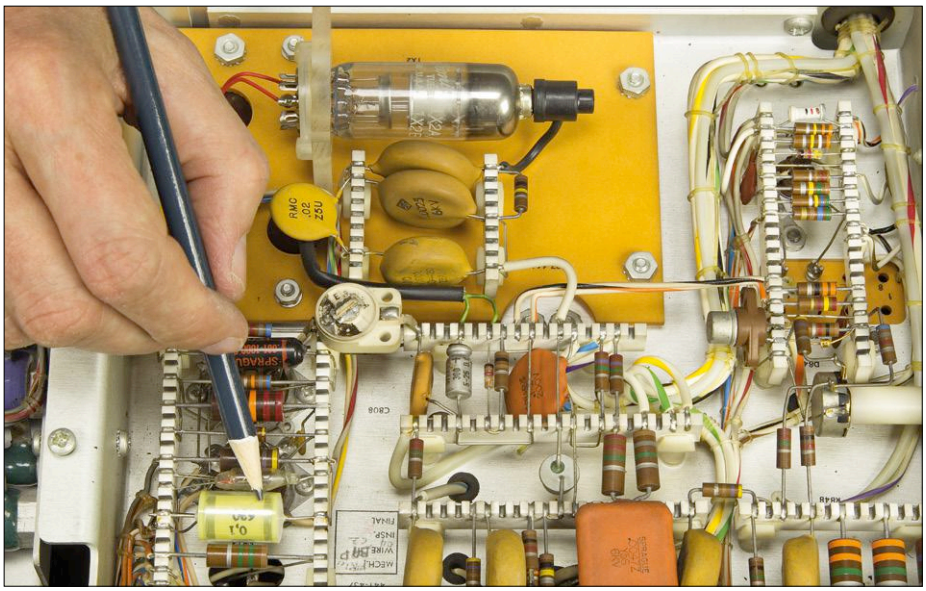


Bild 6. Nach Austausch eines Kondensators funktionierte die Hochspannungsabteilung für die Bildröhre in meinem 565 wieder.

Fazit

Diese Messinstrumente bieten so viele Optionen, dass es schon einmal bis zu 15 Minuten dauern kann, bis die Einstellungen richtig sind und man mit den Messungen anfangen kann. Man bedenke nur einmal, dass hier für zwei Elektronenstrahlen (oben und unten) mit je zwei Kanälen (oben-oben und oben-unten sowie unten-oben und unten-unten) plus zwei Zeitbasen A und B, verzögert und nichtverzögert, zugehörig zu oben und/oder unten, mit Triggerung auf A oder B oder über Kreuz etc. und pp. Einstellungen vorzunehmen sind. Das Ganze untermalt vom Geräusch kräftiger Ventilatoren, die 500 bis 700 W an Wärme aus dem Gehäuse blasen. Und wenn man mal umräumen möchte, braucht man einen kräftigen Freund oder einen guten Orthopäden. Dafür wird man mit einem Design belohnt, das etwas ganz anderes ist als alles, was heute so im modernen Plastikstil geboten wird.

Weder das Modell 565 noch das 556 oder die zugehörigen Einschübe waren kalibriert, wie das nach so vielen Jahren der Vernachlässigung eigentlich erforderlich wäre. Außerdem wäre auch noch eine etwas tiefergehende Reinigung und Optimierung angebracht, doch war mir im Moment die funktionsfähige Präsentation der Geräte für Sie wichtiger. Leider werde ich die nächste Zeit nicht so richtig dazu kommen, damit zu spielen.

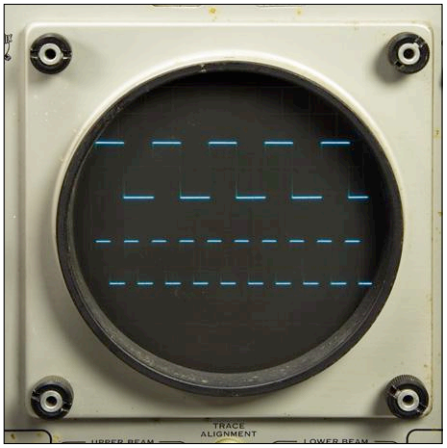


Bild 7. Tektronix-Oszilloskope sind für ihre extrem scharfe Kurvendarstellung und ihr besonderes Impulsverhalten berühmt. Dieses Bild stammt von meinem 565, dessen CRT mit einem feinen Netzgitter als Filter bedeckt war, der zur Schärfe beiträgt. Das Oszilloskop war noch nicht kalibriert.

Sie werden nämlich vorerst in unserem Zentrallager eingemottet, bis der Umzug von Elektor nach Aachen komplett über die Bühne gegangen ist. An die Kollegen im Lager: Seid bitte vorsichtig mit dem Gabelstapler! ⏮

(160402)

ESTD 2004

www.elektor.tv

Retronik ist eine Rubrik, die antiker Elektronik und legendären Elektor-Schaltungen ihre Reverenz erweist. Beiträge, Vorschläge und Anfragen telegrafieren Sie bitte an Jan Buiting (editor@elektor.com).

Sniffer mit dem Radio-Modul RFM12

Belauschen des Funkverkehrs

Von **Laurent Labbe** (China)

Vor einigen Jahren entwickelte ich ein elektronisches Nixie-Röhren-Thermometer, das neben der Uhrzeit und der Innentemperatur auch die Außentemperatur anzeigen konnte, die von einem entfernten Sensor ermittelt wurde. Dazu wurde noch angezeigt, wie oft der Briefträger meinen etwas entfernt vom Haus aufgestellten Briefkasten besucht hatte. Die Sensoren senden ihre Daten in Intervallen von 30 Minuten per Funk zur Haupteinheit des Projekts. Aus verschiedenen Gründen gelang es mir nicht, meinen PC direkt mit diesem Gerät zu verbinden, um herausfinden zu können, ob auch wirklich alle Daten ankommen. Daher habe ich mir ein HF-Empfangsmodul RFM12 zugelegt, um den Funkverkehr zu belauschen.

Mein RFM12-Sniffer empfängt die über HF gesendeten Datenpakete der Thermometer- bzw. Briefkasten-Funksensoren und leitet sie über eine USB-Verbindung an einen PC weiter, der sie anzeigt. Mit der ersten Version meines Sniffers untersuchte ich die Bytes, die von dem oben erwähnten Außen-Thermometer gesendet wurden, wobei das Protokoll einen Header besaß und mit einer Prüfsumme arbeitete. Beim hier gezeigten Projekt ist das Prinzip wesentlich einfacher: Auf dem PC wird alles, was per Funk eintrifft, angezeigt – ohne Untersuchung der Datenstruktur und ohne Prüfsummencheck.

Zwei wichtige Zutaten: PIC und Radiomodul

Die Schaltung ist um den Mikrocontroller PIC18LF2420 aufgebaut (**Bild 1**). Er wird mit 3,3 V versorgt und besitzt eine maximale interne Frequenz von 8 MHz. Es kann jedoch auch jeder andere Mikrocontroller verwendet werden. Der von mir

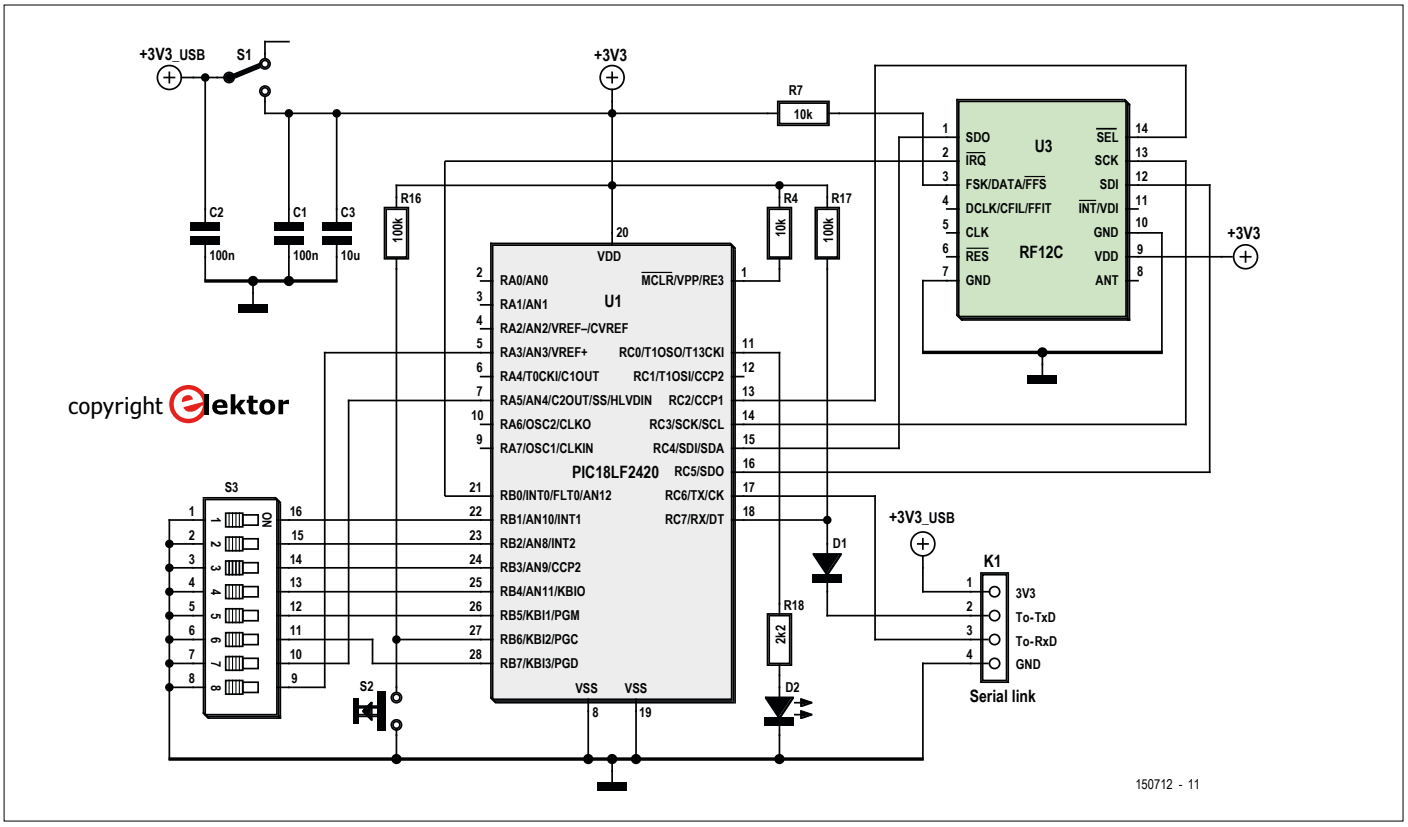


Bild 1. Schaltbild des RFM12-Sniffers.

genutzte USB-Konverter (Prolific) besitzt zwei Versorgungs-Ausgänge: 5 und 3,3 V. Da das RFM12-Modul eine Spannung von 3,3 V benötigt, wird diese verwendet.
Zur Anpassung der Spannungspegel zwischen dem Mikrocontroller (3,3 V) und USB (5 V) ist der Anschluss Tx des Mikrocontrollers direkt mit dem USB-Verbinder K1 verbunden, während der Anschluss Rx mittels einer Diode und einem Pull-Up-Widerstand geschützt wird.

Einstellungen

Mit einer Reihe von DIP-Schaltern (S3) lassen sich verschiedene Einstellungen vornehmen. Die beiden ersten Schalter bestimmen die Übertragungsgeschwindigkeit zum PC (siehe **Tabelle 1**). Wenn am Modul RFM12 nichts mehr ankommt, legen die Schalter 3 und 4 die Zeitspanne fest, bis der PC hiervon eine Rückmeldung über die Zeichen CR/LF bekommt (siehe **Tabelle 2**). Über die Positionen 5 und 6 kann man die Länge der erwarteten Datenframes einstellen (**Tabelle 3**). Achtung: Die Schalter 7 und 8 dürfen nicht benutzt werden. Sie sind an Port A ohne Pull-Up-Widerstände angeschlossen, obwohl Port A selbst keine internen Widerstände besitzt.
S1 (Ein/Aus) dient vor allem zum Neuprogrammieren der Software des Mikrocontrollers, ohne dass dieser hierfür von der Versorgung abgeklemmt werden muss. Mit Taster S2 ist ein Reset der Software möglich.
Bei den Widerständen handelt es sich hauptsächlich um Pull-up-Widerstände: R4 für den Reset des Mikrocontrollers, R16 für den Taster S2 und R17 für die serielle Verbindung am Anschluss K1. Für den DIP-Schalter werden die internen Pull-up-Widerstände von Port B benutzt. Die LED D2 blinkt, wenn Daten ankommen.

Ein bisschen Code

Der Sourcecode wurde mit MikroC geschrieben und kompiliert (Download siehe [2]). Die Prozeduren für die Ansteuerung des RFM12 sind am kritischsten. Ich habe dazu Routinen verwendet, die ich im Internet gefunden habe. Der Code entspricht einer Kurzfassung des in meinem Thermometer verwendeten Codes. Einige Teile dieses Codes werden in der hier beschriebenen Anwendung nicht benötigt.

Tabelle 1. Übertragungsgeschwindigkeit zum PC (S3).

Pos. 1	Pos. 2	Pos. 1	Pos. 2	Pos. 1	Pos. 2	Pos. 1	Pos. 2
OFF	OFF	ON	OFF	OFF	ON	ON	ON
1200 Baud		9600 Baud		19200 Baud		19200 Baud	

Tabelle 2. Dauer der Zeit, bis ein Return zum PC gesendet wird (S3).

Pos. 3	Pos. 4	Pos. 3	Pos. 4	Pos. 3	Pos. 4	Pos. 3	Pos. 4
OFF	OFF	ON	OFF	OFF	ON	ON	ON
kein CR/LF		1 s		10 s		20 s	

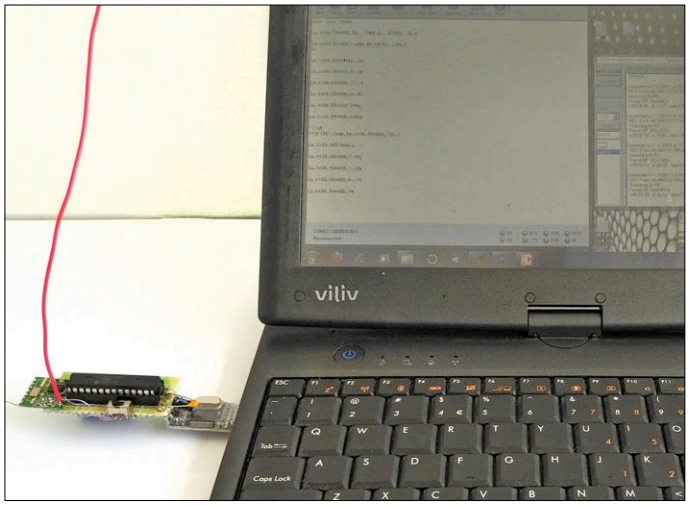


Bild 2. „Schnüffler“ in Aktion:
Die an einen Laptop angeschlossene Schaltung.

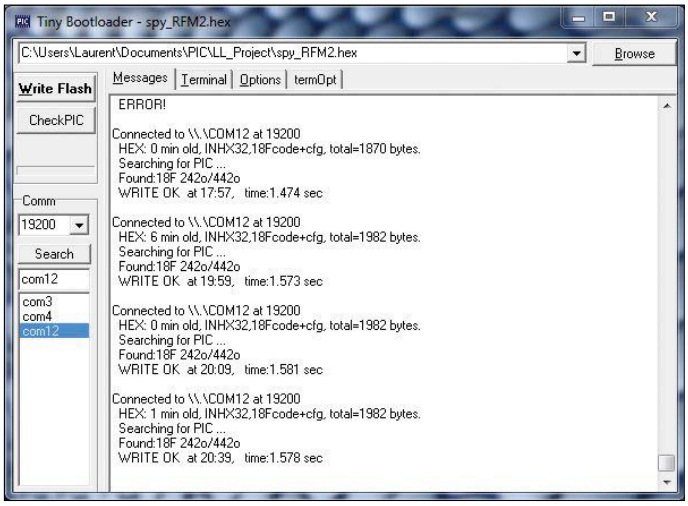


Bild 3. Screenshot des Bootloaders.

In der Hauptschleife wird hauptsächlich gewartet. Der /IRQ-Ausgangspin des RFM12-Moduls meldet dem Mikrocontroller die Ankunft von Daten (Eingang PORTB.F0). Die Daten werden dann weiter zum Mikrocontroller geleitet (PORTC.F4), um über die serielle Verbindung (PORTC.F6) zum PC gesendet zu werden. Ich habe festgestellt, dass der RFM12 oft „fehlerhafte“ Daten sendet. Daher müssen der Empfänger und das FIFO-Register regelmäßig auf Null gesetzt werden. In meinem Fall verwende ich die DIP-Schalter 5 und 6, um die Länge eines Frames einstellen zu können (siehe Tabelle 3). Ist die eingestellte Länge erreicht, werden Empfänger und Register automatisch re-initialisiert.
Normalerweise verwende ich für meine Anwendungen mit dem 18LF2420 einen ganz normalen Bootloader. Den *Tiny Pic Bootloader* (siehe [3]) finde ich dabei am praktischsten, obwohl er nicht zu allen Mikrocontrollern kompatibel ist. Der Code umfasst etwa 100 Bytes und funktioniert sehr zuverlässig.

Daten lesen

Die Anwendung der Schaltung ist einfach: Verbinden Sie die Anordnung mit dem USB-Port eines PCs. Der für das USB-Modul benötigte Treiber muss vorinstalliert sein. Verwenden Sie ein Tool wie zum Beispiel *HyperTerminal* oder *CoolTerm* und stellen Sie die korrekte Geschwindigkeit ein. Die Daten müssen angezeigt werden. Wenn der Timer (Tabelle 2) aktiviert ist, verbessert sich die Lesbarkeit durch den Zeilenumbruch. Mit der Speicher-Funktion können Sie das Gerät während mehrerer Tage betreiben und die Frames dann später auswerten. Mit Cool Term hängt die Anzeige normaler Zeichen manchmal fest, aber wenn die Daten hexadezimal dargestellt werden, gibt es keine Probleme. ◀

(150712)

Weblinks

[1] Projekt auf Elektor /Labs: www.elektormagazine.de/labs/sniffer-for-rfm12-display-exchanges-over-the-air-on-pc

[2] Sourcecode, kompilierter Code, Layout-Vorschlag: www.elektormagazine.de/150712

[3] Tiny Pic Bootloader: www.etc.ugal.ro/cchiculita/software/picbootloader.htm

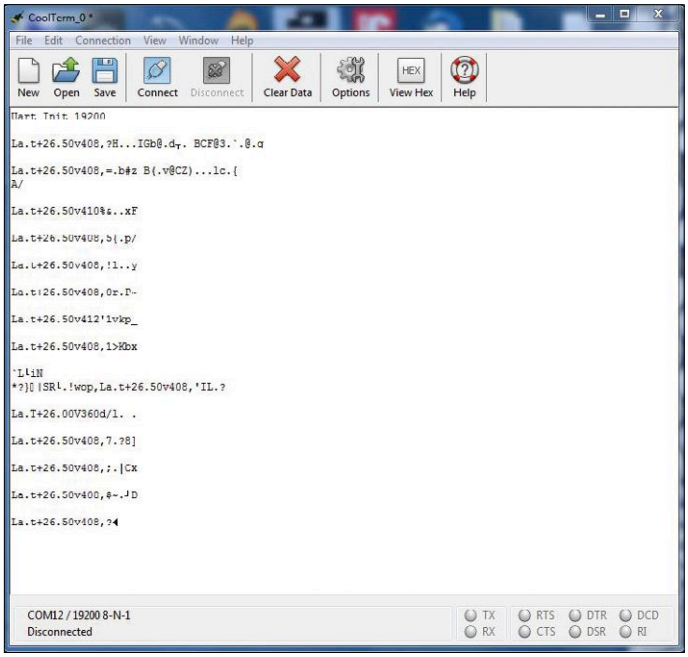


Bild 4. Screenshot des Terminalprogramms.

Tabelle 3. Länge des erwarteten Frames (S3).

Pos. 5	Pos. 6	Pos. 5	Pos. 6	Pos. 5	Pos. 6	Pos. 5	Pos. 6
OFF	OFF	ON	OFF	OFF	ON	ON	ON
10 Bytes		20 Bytes		40 Bytes		100 Bytes	

LoRaWAN

Architektur, Protokoll, Sicherheit und Betrieb

Mein letzter Beitrag [1] beschäftigte sich mit LoRa, eine von Semtech entwickelte drahtlose Netzwerktechnik. LoRa überbrückt weite Distanzen (mehrere 10 km im freien Feld) mit niedrigen Übertragungsgeschwindigkeiten, das Modulationsverfahren hat den Namen *Chirp Spread Spectrum* (CSS). Um LoRa in ein Projekt zu integrieren, genügt das Hinzufügen eines LoRa-Chips oder LoRa-Moduls. In dieser Folge geht es um LoRaWAN, ein auf LoRa aufsetzendes offenes Protokoll, das von der internationalen *LoRa Alliance* administriert wird. Ein wichtiger Aspekt ist auch die Datensicherheit.

Von **Robert Lacoste** (F)

Protokoll

Am Anfang soll zunächst eine allgemeine Betrachtung von Protokollen stehen, die in der Kommunikationstechnik angewendet werden. Als die ersten Netzwerke entwickelt wurden, erwies es sich als zweckmäßig, die Betriebsprotokolle in voneinander unabhängige Bereiche zu unterteilen, sie wurden Schichten oder *Layer* genannt. Ein solches Referenzmodell erleichtert das Standardisieren, Programmieren und Testen, es steigert entscheidend die Flexibilität. Wenn ein Protokoll in unabhängigen Schichten strukturiert ist, lassen sich Modifikationen einer Schicht ohne Auswirkungen auf eine andere Schicht durchführen. Die Unabhängigkeit der Schichten erlaubt es, zum Beispiel die Protokolle HTTP oder FTP transparent in einem Ethernet- oder WLAN-Netzwerk anzuwenden.

Das OSI-Referenzmodell [2], skizziert in **Bild 1**, wurde im Jahr 1984 aus der Taufe gehoben, es ist trotz der Jahre auch heute noch aktuell. Das Modell beginnt von unten nach oben mit der so genannten physischen Schicht, sie ist ausschließlich für die Übertragung der Bits zuständig. Darüber liegt die Sicherungsschicht, die den Informationsaustausch zwischen zwei Punkten regelt, gefolgt von weiteren Schichten bis hin zur Anwendungsschicht. In dieser obersten Schicht erhalten die übertragenen Bits ihren Informationsgehalt. Im OSI-Referenzmodell ist LoRa ganz unten in der physischen Schicht angesiedelt. LoRa definiert, wie die Bits durch CSS-Modulation kodiert und drahtlos übertragen werden. Zwar ist es möglich, für LoRa ein individuelles Protokoll zu schreiben, doch dann müssen zahlreiche Details überdacht und berücksichtigt werden: Selektieren und Adressieren, Behandeln von Fehlern und Datenkonflikten, Sichern des korrekten Datenaustauschs, und so weiter. Deutlich weniger aufwendig gestaltet sich der Einsatz von LoRaWAN, ein so genanntes offenes Protokoll, das auf LoRa aufbaut. Gemessen am OSI-Referenzmodell ist LoRaWAN rudimentär: LoRaWAN definiert in einer ausführlichen zweiten Protokollschicht nur die Verbindung zwischen zwei Punkten einschließlich der Sicherung dieser Verbindung. Eine Anwendungsschicht gehört nicht zu LoRaWAN. Das bedeutet, dass LoRaWAN die Daten von einem Punkt zum anderen überträgt, aber nicht interpretiert.

Systemarchitektur

LoRaWAN ist nicht nur ein Protokoll, sondern auch ein Netzwerkmodell. In **Bild 2** ist die Architektur eines LoRaWAN-Netzwerks in Symbolen dargestellt. Das Netzwerk besteht aus drei Subsystemen: Objekte, Gateways und Server. Ein LoRaWAN-Objekt kann jedes Gerät mit Netzwerkschnittstelle sein, beispielsweise ein vernetzter Rauchmelder. Die LoRaWAN-Objekte senden ihre Nachrichten im Broadcast-Modus, also ohne Adressen von Empfängern. Die Nachrichten werden von einem oder mehreren Gateways empfangen und an den zentralen Netzwerkserver weitergeleitet. Im Gegensatz zu den Objekten basieren die Gateways auf einer komplexen Architektur, die sich

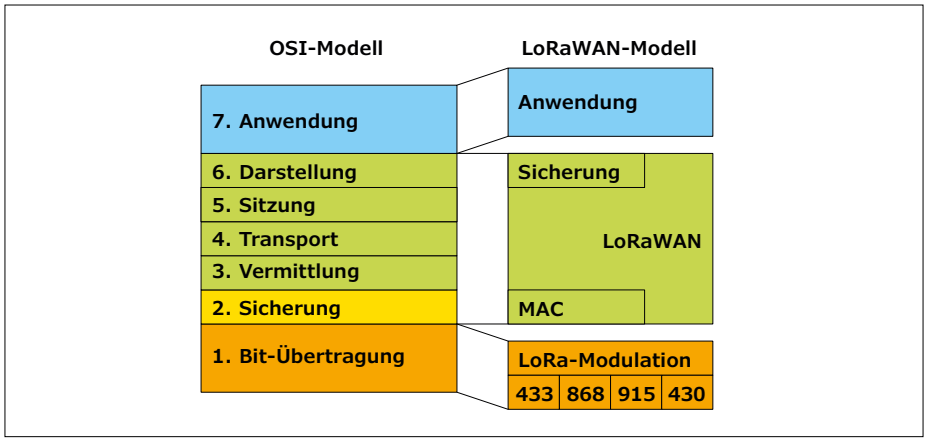


Bild 1. LoRa ist ein Protokoll für die Bitübertragungsschicht, LoRaWAN ist ein darüber liegendes Kommunikationsprotokoll. Die Anwendung ist nicht Bestandteil von LoRaWAN.

um einen spezialisierten Chip von Semtec mit der Typenbezeichnung SX1301 rankt. Ein Gateway muss gleichzeitig auf mehreren HF-Kanälen Signale unterschiedlicher Modulationsarten empfangen, während ein Objekt für nur eine Betriebsart ausgelegt sein muss. Das Entwickeln und Bauen von Gateways gehört zu den Betätigungsfeldern diverser Hersteller, beispielsweise Multitech, Kerlink oder IMST. Ein Gateway lässt sich auch realisieren, indem ein Funkmodul von IMST, das mit dem SX1301 aufgebaut ist, mit einem Raspberry Pi gekoppelt wird. Zur LoRaWAN-Architektur gehören außer den Gateways mehrere zentrale Server, ein Netzwerkserver sowie ein oder mehrere Anwendungsserver. Sämtliche Server können auf einem gemeinsamen Computer untergebracht sein. Die Gateways haben sozusagen die Funktion von Briefkästen, sie empfangen Datenframes und leiten sie über ein 3G- oder ein anderes Netzwerk an einen Netzwerkserver weiter. Der Netzwerkserver erledigt die übrige Arbeit: Er prüft die Inhalte der Frames, entfernt Doubletten und entscheidet, an welchen Anwendungsserver die Information weitergeleitet werden muss. Falls eine Antwort erforderlich ist, veranlasst er ein Gateway, die Antwort an das Objekt zurückzuschicken. Der Netzwerkserver hat noch viele weitere Aufgaben, zum Beispiel das Koppeln von Objekten mit dem Netzwerk, das Sichern des Netzwerks nach außen, die Konfiguration von Netzwerkkomponenten, und so weiter. In kleinen LoRaWAN-Netzwerken arbeiten häufig vereinfachte Server, sie können physisch, wie bei Multitech oder Kerlink, in Gateways integriert sein. Ab einem gewissen Netzwerkumfang ist jedoch ein separater Server nötig. Hersteller oder Vertreiber solcher Server sind beispielsweise Actility, Orbiwise oder Loriot. Parallel dazu existiert eine Open-source-Initiative für einen LoRaWAN-Server auf Github [3]. Der Anwendungsserver verarbeitet die Nutzdaten: In einem Brandmeldesystem ist er es, der ein Abbild der installierten Infrastruktur auf dem Tablet darstellt und im Brandfall den vorgesehenen Alarm auslöst.

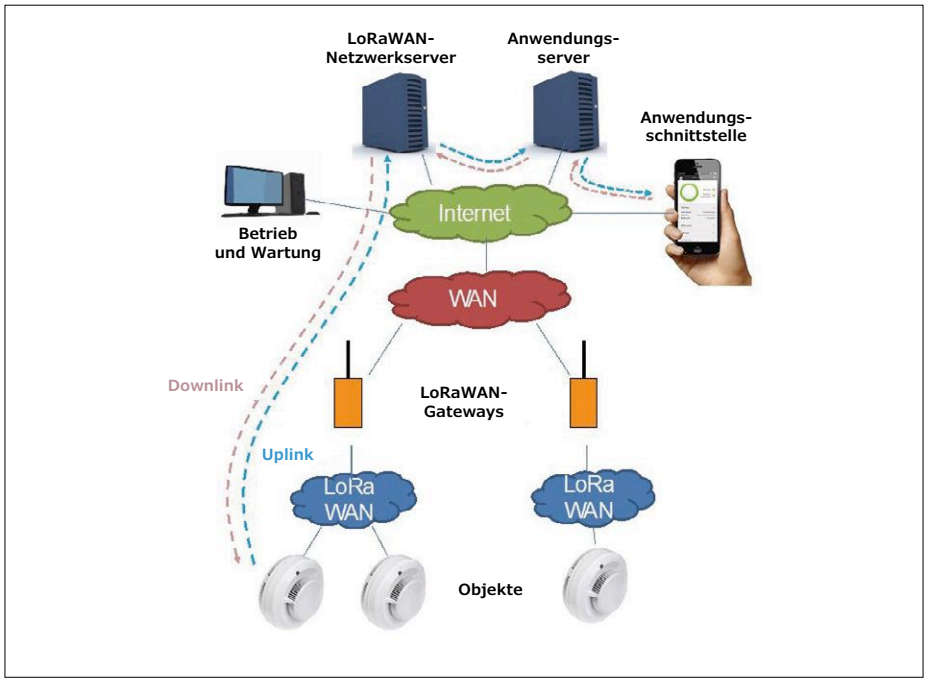


Bild 2. Allgemeine Struktur eines LoRaWAN-Netzwerks.

LoRa Alliance

LoRaWAN wird von einer Organisation administriert und organisiert, die von Semtech unabhängig ist, sie nennt sich *LoRa Alliance*. Auf der Website [4] ist dargelegt, dass die Allianz inzwischen mehr als 400 Mitglieder zählt, darunter natürlich auch Semtech. Auf der Mitgliederliste stehen neben vielen kleineren Unternehmen auch Namen wie IBM, ARM, Orange, Bouygues Telecom, KPN, STMicroelectronics, Microchip, Cisco und Keysight. Das beeindruckt umso mehr, als die Organisation erst seit rund zwei Jahren existiert. Offiziell pflegt die Organisation den LoRaWAN-Standard [5], und sie veröffentlicht die Quellcodes zum Entwickeln von LoRaWAN-Produkten [6]. Nur um Missverständnissen vorzubeugen: Einerseits ist LoRaWAN ein offenes Protokoll, wobei die Spezifikationen und der Quellcode öffentlich zugänglich sind. Andererseits setzt LoRaWAN auf dem LoRa-Modulationsverfahren CSS auf, das der Semtech Corporation gehört und bisher nicht in den Details freigegeben wurde. Die LoRaWAN-Allianz hat außerdem ein Verfahren für die Zertifizierung erarbeitet, die sicherstellen soll, dass Produkte den LoRaWAN-Standard erfüllen und untereinander kompatibel sind. Die Mitgliedschaft in der Allianz und das anschließende Zertifizieren eines Produkts kosten mehrere 1000 €. Die Kosten lassen sich umgehen, wenn ein bereits zertifiziertes LoRa-Modul zum Einsatz kommt, denn dann ist dieser Schritt bereits getan.

Klassen der Endgeräte

Wie läuft der Informationsaustausch zwischen einem Objekt mit Netzwerkanbindung und dem Netzwerk über LoRaWAN ab? Dazu wurden, wie in **Bild 3** skizziert, drei Objektklassen definiert. Die

untere Klasse ist die Klasse A: Das Objekt befindet sich fast ununterbrochen im Sleep-Zustand. Es wacht nur auf, wenn es eine LoRaWAN-Nachricht über einen Funkkanal absetzen möchte. Danach fällt es in den Sleep-Zustand zurück, um eine Sekunde später wieder aufzuwachen und auf dem gleichen Funkkanal nach einer Antwort zu suchen. Wenn die Antwort ausbleibt, wird eine weitere Sekunde später ein definierter Ausweichkanal geprüft. Ist das geschehen, tritt wieder der Sleep-Zustand ein. Wegen der sehr kurzen aktiven Intervalle ist der Energiebedarf eines Klasse-A-Objekts äußerst gering. Allerdings wird dies damit erkauft, dass eine Kommunikation mit dem Objekt nicht zustande kommen kann, solange das Objekt nicht selbst aktiv wird. Anders als ein Klasse-A-Objekt wird ein Objekt der Klasse C aus dem Stromnetz mit Energie versorgt, es ist permanent empfangsbereit. Die Klasse B, die sich zwischen A und C bewegt, wird gegenwärtig noch nicht voll unterstützt. Hier wechselt das Objekt periodisch vom Sleep-Zustand in die aktive Phase, um den Funkkanal zu prüfen. Klasse B macht einen komplexen Ablauf notwendig, denn dabei muss eine Synchronisation zwischen dem Netzwerk und den Objekten stattfinden. Das Netzwerk darf Nachrichten nur während vorgegebener Zeiten senden.

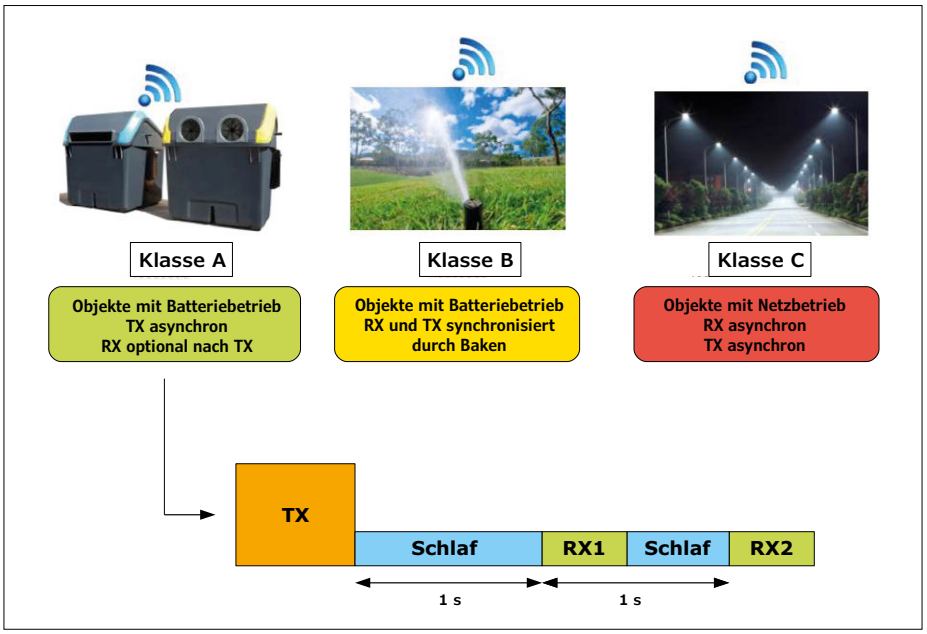


Bild 3. LoRaWAN definiert drei Objektklassen. Die meisten Anwendungen gehören der Klasse A an: Das Objekt ist aktiv aus eigener Initiative, anderenfalls befindet es sich es im Sleep-Zustand.

Variable Datenrate

Wie schon im vorangegangenen Beitrag [1] beschrieben, ist die LoRa-Modulation bezogen auf die Datenrate hoch flexibel. Aus **Bild 4** geht hervor, dass mit niedrigen Datengeschwindigkeiten hohe räumliche Reichweiten erzielbar sind, während hohe Geschwindigkeiten den Energiebedarf und die Belegung des Frequenzbands reduzieren. Um beides gegeneinander zu optimieren, wendet LoRaWAN ein Verfahren an, das *Adaptative Data Rate* (ADR) genannt wird. Auf der Seite des Objekts ist das Verfahren einfach: Der Netzwerkserver fragt beim Objekt an, ob eine bestimme Geschwindigkeit möglich ist. Dazu prüft er, ob das Objekt auf höhere Geschwindigkeiten geschaltet werden kann. Maßgeblich sind die Sendeleistung sowie die Qualität empfangener LoRaWAN-Nachrichten. Anschließend wird das Objekt veranlasst, die Geschwindigkeit an die ermittelten Kriterien anzupassen. Ein anzuwendender Algorithmus ist in LoRaWAN nicht definiert, jeder Netzwerkserver kann mit einem eigenen Algorithmus arbeiten.

An dieser Stelle lässt sich bereits erahnen, welche Probleme bei ADR auftreten können. Angenommen die Empfangsqualität der Nachrichten eines Klasse-A-Objekts ist hervorragend, so dass der Server das Objekt auf hohe Geschwindigkeit schaltet. Wenn jetzt ein räumliches Hindernis zwischen Objekt und Gateway den Funkverkehr zum Erliegen bringt, kann der Server die Nachrichten des Objekts nicht mehr empfangen. In Gegenrichtung kann der Server das Objekt nicht mehr veranlassen, die Geschwindigkeit zu drosseln, damit die Empfangsqualität steigt. Wie schon beschrieben, sind Klasse-A-Objekte nur empfangsbereit, nachdem sie aus eigener Initiative eine Verbindung aufgebaut haben. Deshalb müssen Klasse-A-Objekte, die mit ADR arbeiten, von Zeit zu Zeit mit niedriger Geschwindigkeit eine Bestätigung des ordnungsgemäßen Betriebs anfordern.

Sicherheit hat Vorrang

Es liegt in der Natur der drahtlosen Kommunikation, dass ungeschützte Informationen unbemerkt von Unbefugten mitgelesen werden können. Maßnahmen zur Sicherung gegen Hacker, Spione oder Saboteure sind deshalb unabdingbar. Der Grad der Geheimhaltung und der Umfang der Maßnahmen hängen natürlich von der Anwendung ab. Für brisante finanzielle Transaktionen gelten andere Maß-

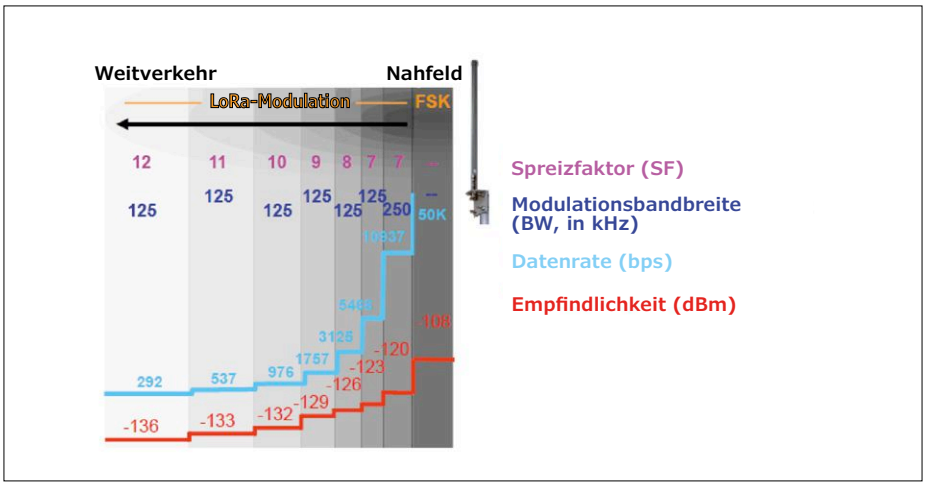


Bild 4. Mit ADR (Adaptative Data Rate) passt der Netzwerkserver dynamisch die Übertragungsgeschwindigkeit an, sie hängt von der Übertragungsqualität ab.

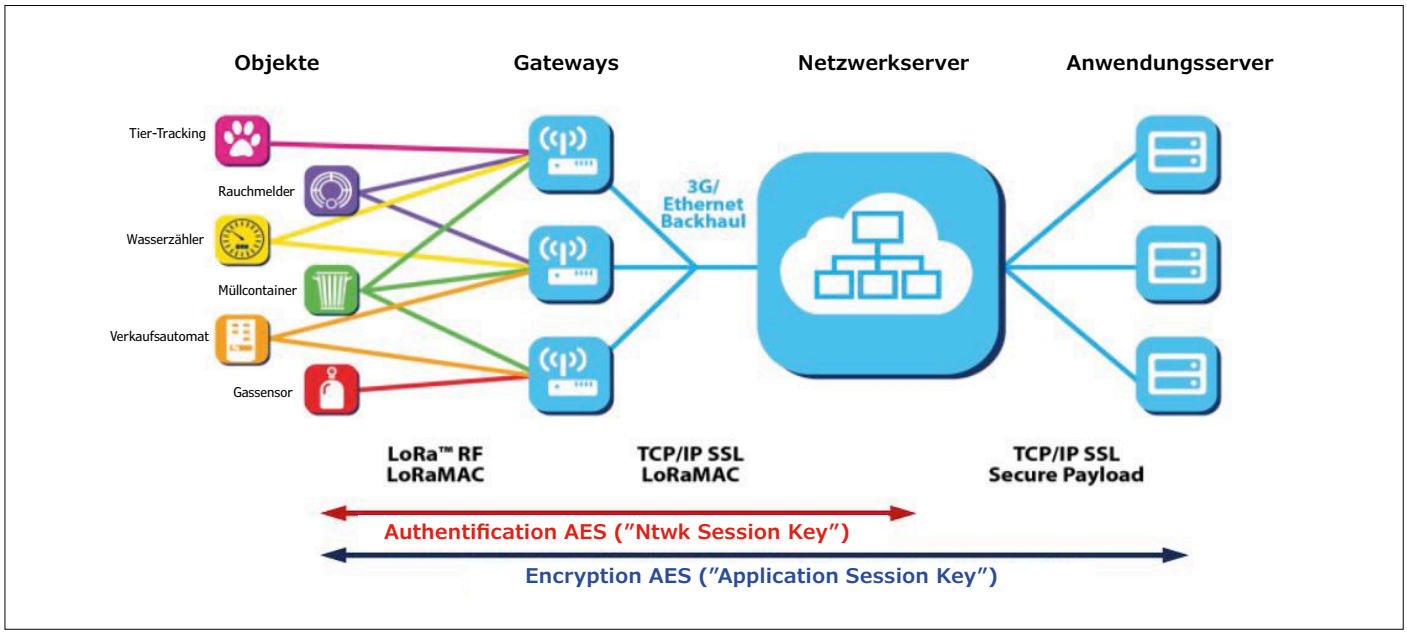


Bild 5. In LoRaWAN existieren zwei Sicherungsebenen: Authentifizierung zwischen Objekten und Netzwerkserver sowie End-to-end-Verschlüsselung zwischen Objekt und Anwendungsserver.

stäbe als beispielsweise für den Tracker, der den Weg eines Haustiers verfolgt. In LoRaWAN sind Sicherungsmechanismen implementiert, die für 99 % der Anwendungen ausreichen. Das Konzept sieht von Anfang an vor, dass mehrere voneinander unabhängige Anwendungen im gleichen Netzwerk unterstützt werden. Deshalb sind in LoRaWAN, wie in **Bild 5** dargestellt, zwei Sicherungsebenen eingebaut. An erster Stelle steht die Verschlüsselung der Nachrichten durch den Absender. Das geschieht mit einem geheimen Schlüssel, dem ***Application Session Key***, er ist ausschließlich dem Objekt und dem zugeordneten Anwendungsserver bekannt. Weil es sich um eine Verschlüsselung zwischen zwei Endpunkten handelt, können dazwischen liegende LoRaWAN-Netzwerkknoten den Inhalt nicht mitlesen.

Anschließend wird die Nachricht mit einer digitalen Signatur unterschrieben. Die Signatur wird mit dem ***Network Session Key*** generiert, einem zweiten geheimen Schlüssel; ihn kennen nur das Objekt und der Netzwerkserver. An der Signatur kann der Empfänger erkennen, dass die Nachricht unterwegs nicht verfälscht wurde (Integrität) und dass sie tatsächlich von dem darin genannten Absender stammt (Authentizität). Der Prozess der Sicherung ist schematisch in **Bild 6** dargestellt. Zwei Sicherungsebenen sind notwendig, weil überwiegend mehrere Anwender eine gemeinsame LoRaWAN-Infrastruktur nutzen. Der Anwender verhindert, dass der Infrastruktur-Betreiber Nachrichten mitlesen kann, und dieser stellt sicher, dass er nur unbeschädigte Nachrichten transportiert. Die beiden Sicherungsebenen wurden mithilfe von zwei Varianten der AES-Verschlüsselung entwickelt. Es handelt sich um ein symmetrisches Verschlüsselungsverfahren, bei dem die Schlüssel den Netzknoten von Anfang an bekannt sein müssen. Für LoRaWAN kann die Vergabe bei der Geräteproduktion geschehen, indem die Schlüssel zusammen mit einer individuellen Kennung in die Objekte eingebrannt werden, gleichzeitig werden die Schlüssel in den Servern hinterlegt.

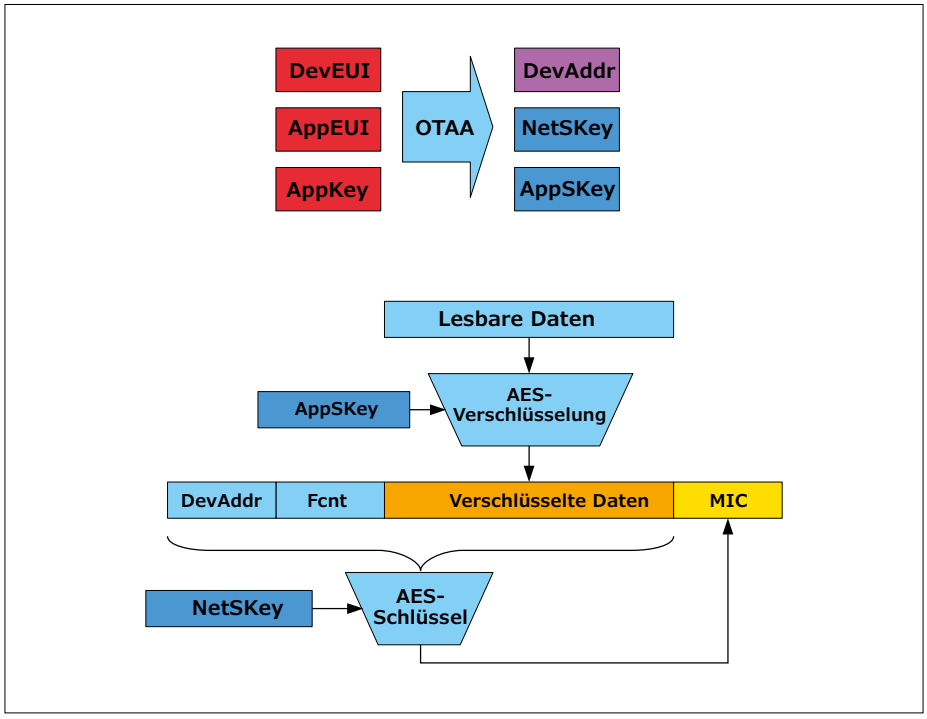


Bild 6. Zur Sicherung sind zwei Schlüssel und eine Netzwerkadresse nötig. Diese Daten können mit OTAA zum Objekt übertragen werden.

Eine andere Methode der Schlüsselübergabe heißt *Over The Air Activation* (OTAA), sie ist deutlich flexibler, weil hier ein Objekt das Netzwerk bei Bedarf wechseln kann. Die Objekte werden mit einer individuellen Kennung und einem anderen geheimen Schlüssel programmiert, der ausschließlich für die Kopplung mit dem Netzwerk gültig ist. Dieser Schlüssel hat die etwas unglückliche Bezeichnung *Application Key* erhalten, er darf nicht mit dem *Application Session Key* verwechselt werden.

Wenn ein Objekt in Betrieb genommen wird, startet es mit einer Anfrage an alle erreichbaren Netzwerke mit der Bitte um Zugang. Gewährt ein Netzwerk den Zugang, generiert es die anderen beiden Schlüssel und sendet sie zum Objekt. Die Übertragung findet verschlüsselt und authentifiziert statt, die Schlüssel werden niemals mitlesbar übertragen. Diese Prozedur ist zwar komplexer, doch sie dient der Sicherheit.

Links und Quellen

[1] LoRa, Niedrige Geschwindigkeit, hohe Reichweite, Elektor, Mai/Juni 2017: www.elektormagazine.de/160236

[2] OSI-Modell: https://de.wikipedia.org/wiki/OSI-Modell

[3] Open source LoRaWAN-Server-Software: http://github.com/brocaar/loraserver

[4] LoRa Alliance: www.lora-alliance.org

[5] LoRaWAN-Standard 1.02: http://lora-alliance.org/Contact/Request-Specification-Form

[6] Referenz-Quellcode für LoRaWAN: http://github.com/Lora-net/LoRaMac-node/wiki/LoRaMAC-node-Wiki

LoRaWAN-Betreiber

Um LoRa und LoRaWAN in Projekte einzubinden, stehen drei Optionen zur Wahl. Die erste Option ist der Erwerb integrierter Bausteine oder Module, die mit LoRa kompatibel sind. In diesem Fall wird LoRaWAN nicht zwingend benötigt, es genügt ein kleines, eigenes Protokoll. Die Lösung ist dann perfekt, wenn zum Beispiel eine Fernbedienung für ein Garagentor mit einer Reichweite von mehreren Kilometern realisiert werden soll. Ob dies immer sinnvoll ist, soll an dieser Stelle außer Betracht bleiben.

Die zweite Option ist der Aufbau eines privaten LoRaWAN-Netzwerks. Dazu müssen ein oder mehrere Gateways sowie ein Netzwerkserver beschafft werden. Die Objekte, in denen der LoRaWAN-Stack implementiert sein muss, sind zu entwickeln oder käuflich zu erwerben. Erst wenn das geschehen ist, können die Objekte LoRaWAN nutzen und Nachrichten über das Netzwerk mit dem Anwendungsserver austauschen. Diese Option ist wesentlich aufwendiger, aber auch flexibler, sie kann sinnvoll für ein privates Kommunikationssystem sein, das beispielsweise in einem Produktionsbetrieb oder einem Krankenhaus installiert ist.

Die dritte und gleichzeitig gebräuchlichste Option besteht darin, die Dienste eines LoRaWAN-Netzwerkbetreibers zu nutzen. Die etablierten Anbieter klassischer Telekommunikation haben oft auch weitläufige LoRaWAN-Netzwerke installiert. Bei dieser Option müssen nur LoRaWAN-kompatible Objekte und, falls nötig, die zugehörigen Anwendungsserver entwickelt werden. Das Nutzen eines kommerziellen LoRaWAN-Netzwerks ist natürlich mit Kosten verbunden. Inzwischen existieren weltweit einige Dutzend Betreiber, hier sollen nur exemplarisch folgende Namen genannt werden: Bouygues Telecom (mit der Tochter Objenious) und Orange in Frankreich, Proximus in Belgien, KPN in den Niederlanden, Senet in den USA, Tata in Indien. In ihren Einzugsgebieten haben diese Unternehmen fast flächendeckend Gateways installiert, sogar in Gebäuden.

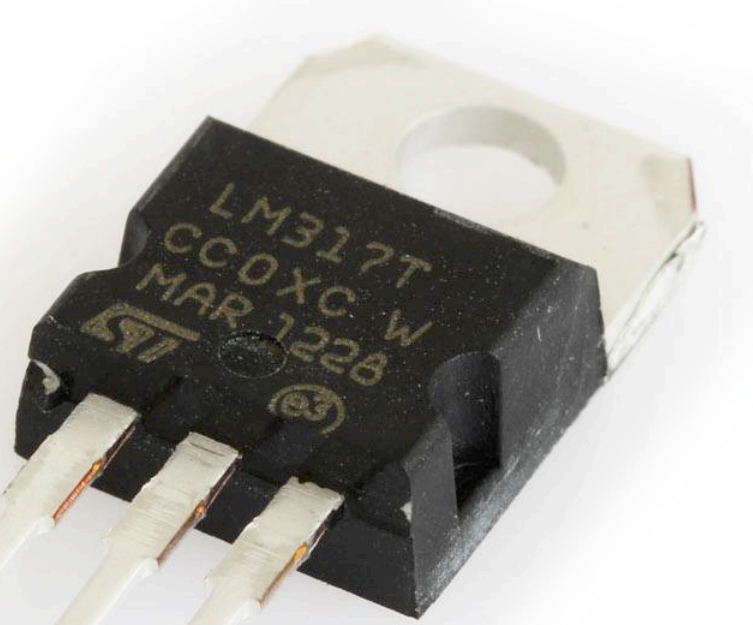
Und die Kosten? Leider sind die LoRaWAN-Tarife nur selten transparent, doch die Entgelte dürften sich auf wenige Euro pro Objekt und Jahr belaufen. Für private Einzelanwender ist der Zugang manchmal schwierig, denn im gegenwärtigen Stadium werden die Großkunden bevorzugt, die Profit versprechende Projekte mit massenhaft vielen Objekten an das Netz anbinden möchten. Solchen Kunden stehen häufig kostenlose Testzugänge offen, die unentbehrlichen Entwicklungs-Kits gehören ebenfalls zum Dienst am Kunden.

Ausblick

Wird LoRaWAN zum Flaggschiff des IoT der nächsten zwanzig Jahre aufsteigen? Niemand weiß das so genau, denn auf diesem Terrain gibt es enorm viel Bewegung. Alternativen wie NB-IoT kommen in Fahrt, und Konkurrenten wie Sigfox sind entschlossen, ihre Linien zu verteidigen. Bremsend könnte sich eine Überlastung des 868-MHz-Bands auswirken, falls die Anzahl der Objekte mit Netzwerkanbindung so rasant wächst, wie die Experten prophezeien. LoRaWAN, so viel ist sicher, kann mit seiner ausgefeilten Technik punkten, die Fortentwicklung verläuft gegenwärtig in Schüben. Schon heute haben namhafte Anbieter flächendeckende Netzwerke mit zahllosen Gateways installiert, Startup-Unternehmen entwickeln immer neue kompatible Objekte und die Anzahl privater LoRaWAN-Netzwerke in der Industrie steigt. Für LoRaWAN wurden vielversprechende Entwicklungen in naher Zukunft in Aussicht gestellt, zum Beispiel die Geolokalisierung von Objekten und internationales Roaming. Unsere Empfehlung lautet: Bleiben Sie am Ball! ◀

(160308)gd

Dieser Beitrag ist auch in Circuit Cellar Nr. 315 erschienen.

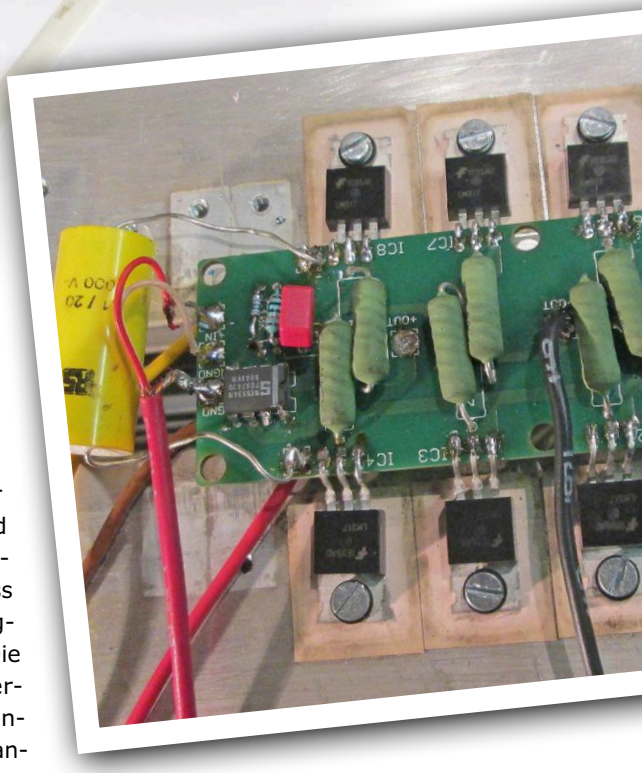


Audioverstärker Klasse A

Unkonventionell mit Spannungsreglern LM317

Von **Jurgen Michiels** (B)

Wenn es nicht auf qualitative Höhenflüge bei den Verzerrungen, beim Frequenzgang und anderen einschlägigen Kriterien ankommt, kann der alternative Entwurf eines Audioverstärkers zu ebenso erstaunlichen wie lehrreichen Ergebnissen führen. Hier wurde der Spannungsregler LM317 zweckentfremdet. Was dabei herauskam, darüber berichten wir hier.



Der Spannungsregler LM317 ist ein Veteran unter seinesgleichen, er kam im Jahr 1976 auf den Markt. Für ihn ist charakteristisch, dass die Ausgangsspannung immer um 1,25 V höher liegt als die Spannung am Anschluss ADJ (Adjust). Während über diesen Anschluss lediglich 100 µA fließen, kann der Ausgangsstrom bis 1,5 A betragen. Der LM317 ist folglich ein Verstärker; wie wir sehen werden, kann er auch Audiosignale verstärken.

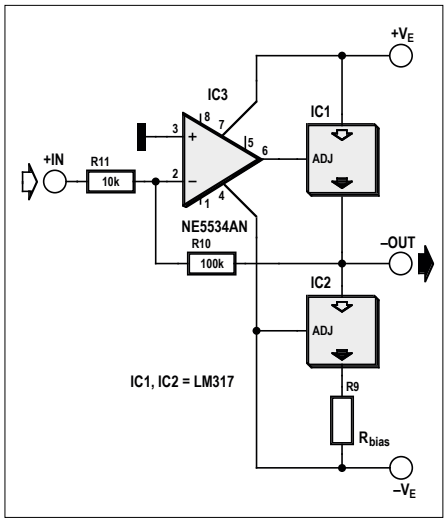


Bild 1. Prinzip des Klasse-A-Audioverstärkers mit zwei LM317.

Konzept

Das Prinzip unseres Audioverstärkers ist in **Bild 1** skizziert. IC1, der erste LM317, wird unmittelbar von Opamp IC3 gesteuert, während IC2, der zweite LM317, als Stromquelle geschaltet ist. Er bewirkt, dass Strom während der negativen Signalhalbwelle zurückfließen kann. Die Höhe des Stroms hängt von Widerstand R9 ab. Da die Ausgangsspannung stets um 1,25 V über der Spannung am Anschluss ADJ liegt, beträgt der Strom 1,25 V / R9. Die Verstärkung wird von R11 und R10 bestimmt. Infolge der Rückkopplung kompensiert IC3 die Offset-Spannung 1,25 V, so dass sich die mittlere Ausgangsspannung auf 0 V einpendelt.

Wie wir sehen, genügen zwei LM317, ein Opamp und drei Widerstände, um einen simplen Klasse-A-Audioverstärker zu bauen. Übrigens: Der Signalweg ist frei von Kondensatoren!

Schaltung

Da der LM317 Ströme bis lediglich 1,5 A verträgt, hält sich die Audio-Ausgangsleistung in Grenzen. Dem lässt sich abhelfen, indem mehrere LM317, wie **Bild 2** zeigt, parallel geschaltet werden.

Die Obergrenze der Eingangsspannung hat der Hersteller beim LM317 auf 40 V festgelegt, sie beträgt bei symmetrischer Spannung ±20 V. Der Opamp hat schon bei ±18 V die Grenze seiner Betriebsspannung erreicht. Als Betriebsspannung unseres Verstärkers haben wir deshalb ±15 V gewählt, diese Spannung muss übrigens nicht stabilisiert sein.

Wir wollen nun den nötigen Ruhestrom bestimmen. Bei der Last 8 Ω beträgt der maximale Strom theoretisch

$$15\ \text{V} / 8\ \Omega = 1{,}875\ \text{A}$$

Die maximale Dauerleistung beträgt dann, ebenfalls theoretisch, ungefähr 14 W, was in der Praxis gute 12 W sind.

Allerdings gilt dies nur bei rein ohmscher Last am Ausgang. Weil Lautsprecher auch eine induktive Komponente haben, setzen wir den Ruhestrom etwas höher (auf 2,2 A) an. Widerstand R9, der den Ruhestrom bestimmt, erhält den Wert

$$1{,}25\ \text{V} / 2{,}2\ \text{A} = 0{,}56\ \Omega$$

Dieser Widerstand setzt knapp 3 W in Wärme um, so dass eine 5-W-Ausführung an dieser Stelle genügt.
Die Verlustleistung unserer Verstärker-Endstufe beträgt insgesamt

$$30\ \text{V} \cdot 2{,}2\ \text{A} = 66\ \text{W}$$

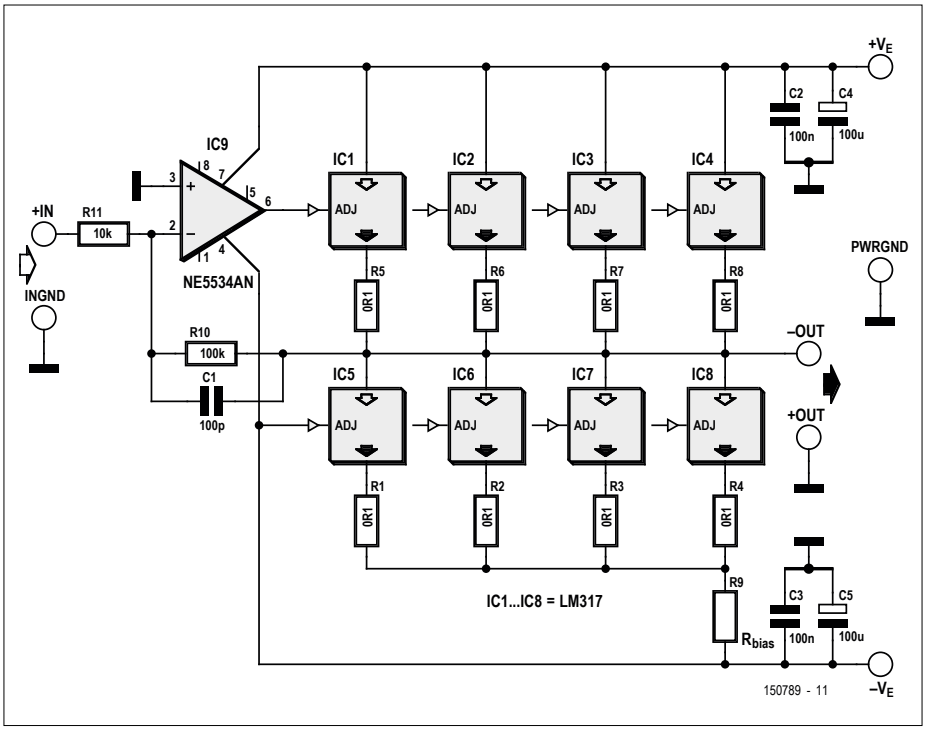
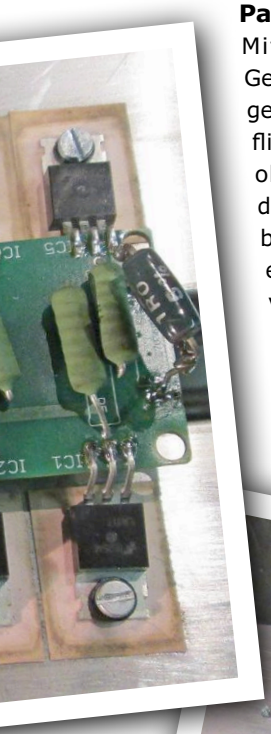


Bild 2. Erst auf den zweiten Blick ist hier ein Audioverstärker erkennbar.

Parallel

Mit vier LM317 darf der Gesamtstrom bis 6 A betragen. Beim Ruhestrom 2,2 A fließen bis 4,4 A durch die oberen LM317 und 2,2 A durch die unteren, was problemlos machbar ist. Bei einer Last 4 Ω am Ausgang verdoppelt sich der Strom, so dass vier LM317 nicht genügen, es sei denn, die Betriebsspannung wird herabgesetzt.



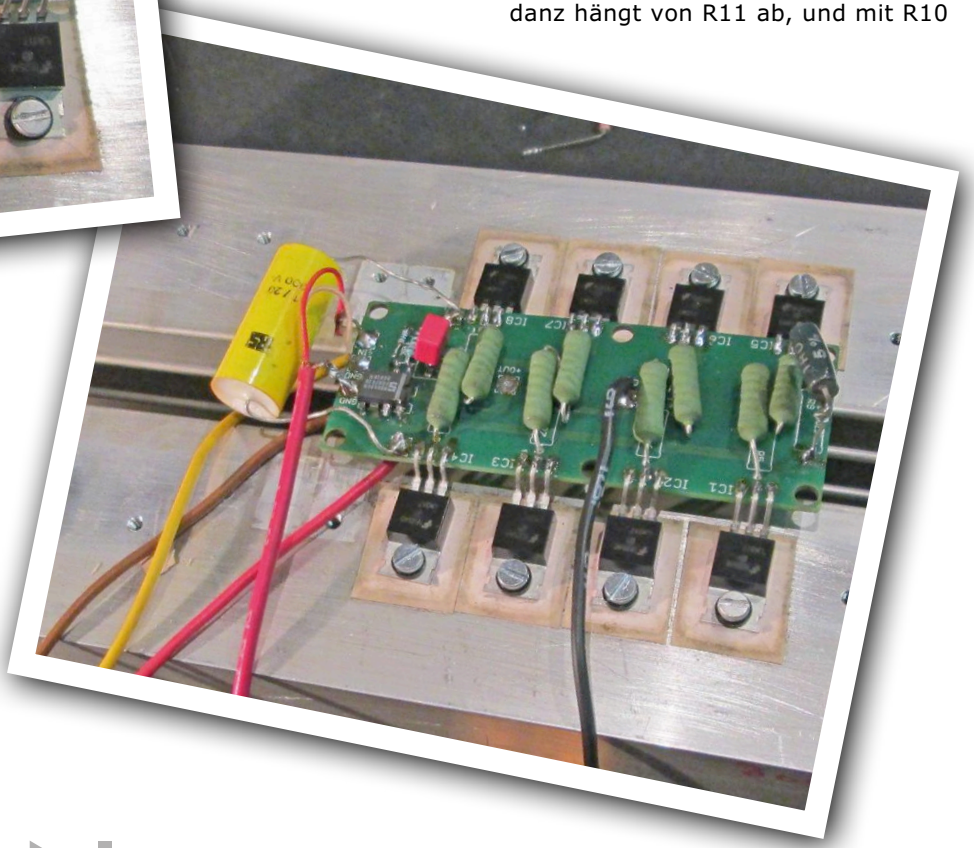
In unserem Verstärker sind jeweils vier LM317 über *Current-sharing*-Widerstände parallel geschaltet, damit sich der Strom über die einzelnen Exemplare möglichst gleichmäßig verteilt. Die Eingangsimpedanz hängt von R11 ab, und mit R10 kann die Verstärkung A festgelegt werden. Hier gilt folgende Beziehung:

$$A = R10 / R11$$

Wir haben die Verstärkung 10 gewählt, R11 ist 10 kΩ und R10 ist 100 kΩ. Kondensator C1 begrenzt die Bandbreite auf etwa 100 kHz und beugt so eventueller hochfrequenter Schwingneigung vor.

Für die Eigenschaften unseres Verstärkers ist in erster Linie der Opamp verantwortlich. Mit dem NE5534 liegen die Verzerrungen (THD) bei 0,05 %. Anzumerken ist noch, dass das Verstärker-Ausganssignal invertiert ist, so dass der „Plus"-Anschluss des Lausprechers mit Masse und der „Minus"-Anschluss mit den Ausgängen der LM317 verbunden werden muss. Die zum Entwurf gehörenden Dateien (Eagle-Format) und das Platinenlayout sind auf der Projektseite [1] als Download verfügbar.

(150789)gd



Weblinks

[1] www.elektormagazine.de/150789

[2] http://vintageaudioprojects.be

Ohne Kondensatoren im Signalweg

Ein fast universeller IR-Sender

Diese missliche Situation ist alt vertraut und weit verbreitet: Zu jedem Audio-, Video- oder anderen elektronischen Gerät gehört eine eigene Fernbedienung. Manchmal füllen die vielen handlichen Kommandogeber sogar großräumige Schubladen. Die dazu gehörenden Geräte bleiben im Dauerbetrieb auf Standby, rund um die Uhr, Tag für Tag, und verschwenden kostbare Energie.

Von **Goswin Visschers** (NL)

Das Problem der diversen unterschiedlichen Fernbedienungen kann ich natürlich lösen, indem ich eine universelle, programmierbare Fernbedienung anschaffe. Doch ich brauche ein Modell mit digitalem Eingang, um alle Geräte gleichzeitig ein- oder ausschalten zu können. Das soll in dem Moment geschehen, in dem eine Person den Raum betritt oder verlässt. Für diesen Zweck wurden zwar die Bewegungsmelder erfunden. Doch da sie mit Fernbedienungen nicht kombinierbar sind, bin ich auf die Idee gekommen, die ich hier vorstelle.

Fast universell

Der Grund, weshalb ich meinen Entwurf „fast universell" genannt habe, ist der Umstand, dass nicht alle existierenden IR-Protokolle in vollem Umfang unterstützt werden können.

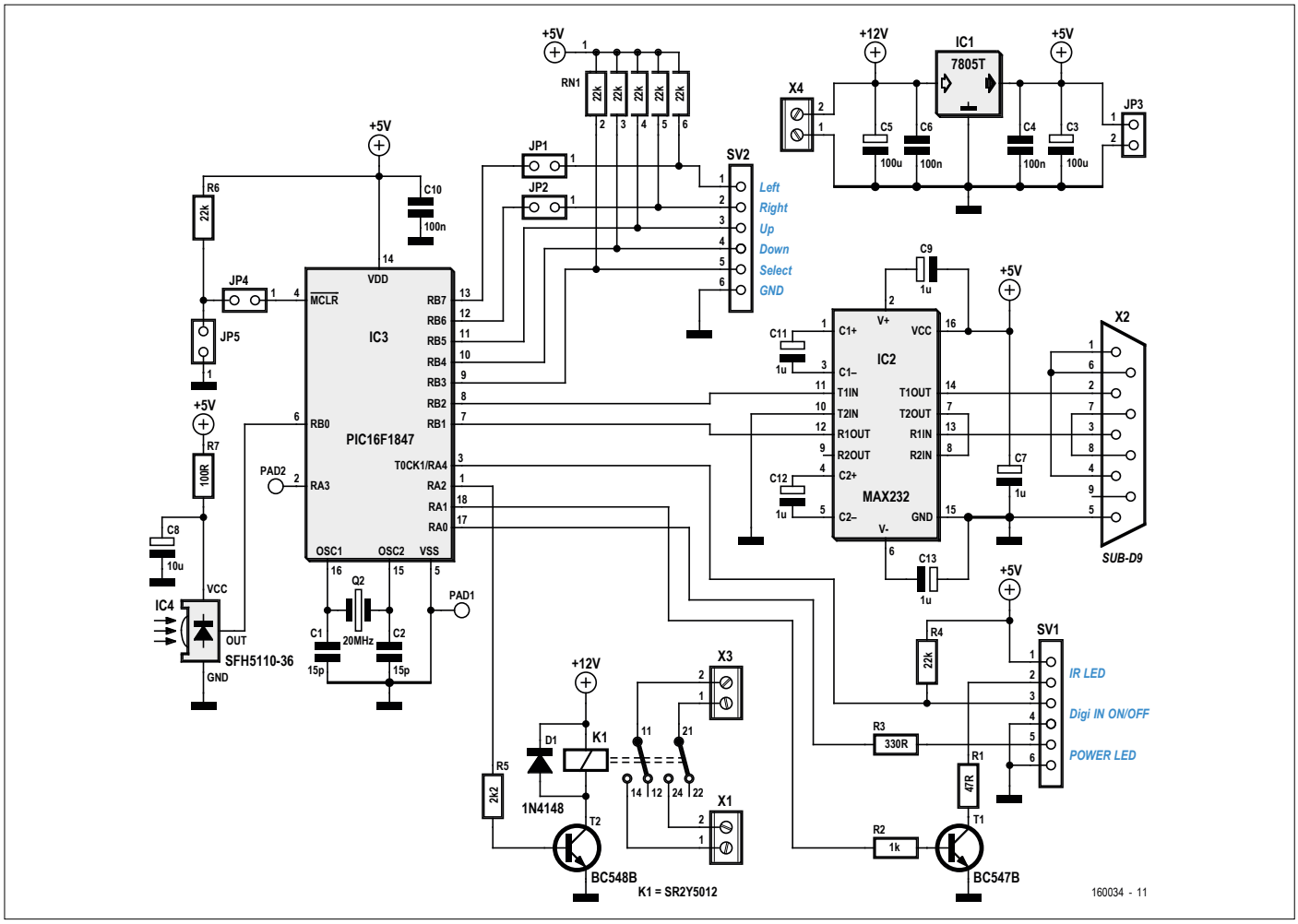


Bild 1. Schaltung des (fast) universellen IR-Senders.

Eigenschaften:

- Unterstützung folgender Infrarot-Fernbedienungssysteme:
  - Philips RC5, RC5-extended, RC6 Mode 0 mit Toggle-Bit
  - NEC -Protokoll und NEC-extended-Protokoll
  - Sony Sirc 12 bit, 15 bit und 20 bit (letztes nicht getestet)
- Steuern der Ein- und Ausschaltcodes über digitalen Eingang
- Parallelschalten mehrerer IR-LEDs möglich
- Einrichten über RS232-Schnittstelle und Terminalprogramm
- Relais zum vollständigen Trennen der ferngesteuerten Geräte vom Stromnetz
- Schalten zwischen vier Gerätegruppen (TV, Radio, CD, USB usw.) mit „Select"-Taster
- Ein- und Ausschalten von Geräten mit bis zu sechs unterschiedlichen IR-Codes
- Vier Taster frei und für jedes Gerät unterschiedlich programmierbar

Die implementierten Versionen habe ich unter „Eigenschaften" zusammengefasst.

Die Schaltung

Aus **Bild 1** ist die Schaltung meines (fast) universellen IR-Senders ersichtlich, sie weist keine ungewöhnlichen Besonderheiten auf. Die Betriebsspannung 5 V wird von einem Spannungsregler 7805 stabilisiert. Ein Pegelwandler MAX232 stellt die Brücke zwischen RS232- und TTL-Signalen her, während ein IR-Empfangsmodul die Signale der originalen Fernbedienungen beim Programmieren an den Mikrocontroller weiterleitet.
Im IR-Empfangsmodul SFH5110-36 ist ein 36-kHz-Filter integriert, Fernbedienungen von Sony arbeiten jedoch mit 40 kHz und Versionen von NEC mit 38 kHz. Das stellt jedoch kein Problem dar, wenn die originale Fernbedienung beim „Lernen" in unmittelbarer Nähe des Moduls positioniert wird. Die Signale kommen dann genügend stark an, um das Filter zu überwinden.
Der Mikrocontroller PIC16F1847 (IC2) ist die zentrale Schaltstelle, er wird von Quarz Q2 mit 20 MHz getaktet. Die Jumper JP1, JP2, JP3 und JP4 sind ausschließlich dazu bestimmt, den Mikrocontroller zu programmieren. Wenn das nicht nötig ist, können JP1, JP2 und JP4 durch Drahtbrücken ersetzt werden. Über Jumper JP3 wird ein In-circuit-Programmer mit der Betriebsspannung versorgt, er kann eventuell entfallen. Mit JP5 wird festgelegt, welches logische Niveau am Eingang Digi IN als aktiv gelten soll. Wenn auf JP5 ein Jumper gesteckt ist, wird Low als aktiver Zustand gewertet, und die Infrarot-LED sendet den Einschaltcode, falls die Kontakte SV1-3 und SV1-4 miteinander verbunden werden. Bleibt JP5 offen, verhält sich Digi IN invertiert.
Die Anschlüsse PAD1 und PAD2 haben hier zunächst keine Funktion. In der Software kann eine Funktion implementiert werden. Relais K1 schaltet beim Senden des Einschaltcodes ein, beim Senden des Ausschaltcodes fällt es ab.
Die Infrarot-LED muss mit SV1-1 und SV1-2 verbunden werden. Wenn mehrere LEDs parallel geschaltet werden sollen, muss Widerstand R4 durch eine Drahtbrücke ersetzt werden, jede LED erhält dann einen eigenen Strombegrenzungswiderstand, beispielsweise 47 Ω.
Die Kontakte SV1-5 und SV1-6 können eine grüne LED steuern. Die grüne LED zeigt durch Blinken die Betriebsbereitschaft an. Die Widerstände des Arrays RN1 ziehen die Eingänge RB3... RB7 des Mikrocontrollers auf High, solange die Taster nicht betätigt werden.

Für die Schaltung habe ich mit Eagle eine Platine entworfen. Die zugehörigen Dateien können von der Projektseite [1] heruntergeladen werden.

Die Software

Die meisten universellen Fernbedienungen „lernen" die Befehle der zu bedienenden Geräte, indem sie die zugehörigen Codes scannen und im Speicher des Mikrocontrollers ablegen. Das hat den Vorteil, dass das Infrarot-Protokoll eine beliebige Version sein kann. Ein Nachteil ist der relativ hohe Speicherbedarf. Die begrenzte Kapazität der EEPROMs der kleinen PIC16-Mikrocontroller von Microchip, die ich bevorzuge, setzt dem Umfang enge Grenzen.
Die Speicherkapazität lässt sich mit einem externen EEPROM erweitern. Ich bin einen anderen Weg gegangen und habe mich entschlossen, die Software des Mikrocontrollers „smarter" zu gestalten. Allerdings wird das dadurch erkauft, dass die Software nur eine begrenzte Anzahl von Protokollen unterstützt. Meine Software kann von der Projektseite [1] heruntergeladen werden.
Die „smarte" Software erkennt das Infrarot-Protokoll, das ihr angeboten wird. Deshalb muss der Mikrocontroller nur eine Kennung des Protokolls und die dazugehörigen Daten speichern. Die Kennung kann als ein Byte gespeichert werden, und die höchstens 32 bit der Daten passen in vier Bytes. Da diese Methode mehr Programmcode als das erste Verfahren erzeugt, habe ich mich für den Einsatz des PIC16F1847 entschieden. Dieser Mikrocontroller hat 14 KB Programmspeicher an Bord. Der Programmspeicher ist vollständig belegt, da möglichst viele Protokolle unterstützt werden sollen.
Ein Protokoll wird erkannt, indem die Breite des ersten Infrarot-Impulses gemessen wird. Bei Hersteller NEC ist der erste Impuls 9 ms breit (**Bild 2**), und der IR-Standard RC5 arbeitet mit einem Startimpuls von 0,9 ms Breite (**Bild 3**). Das Proto-

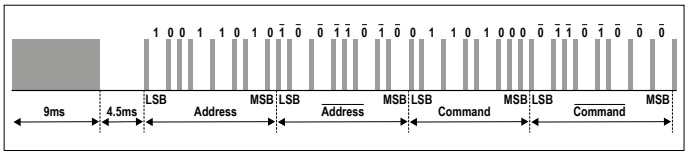


Bild 2. Beispiel für einen IR-Code nach NEC-Protokoll.

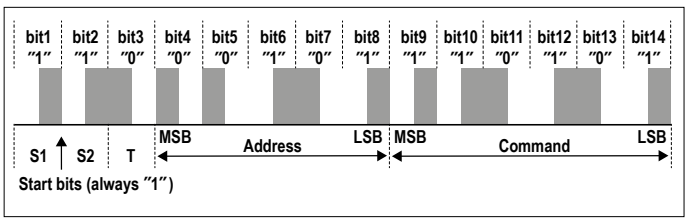


Bild 3. Beispiel für einen IR-Code nach RC5-Standard.

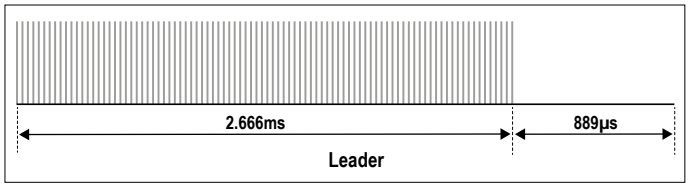


Bild 4. Startimpuls nach RC6-Standard.

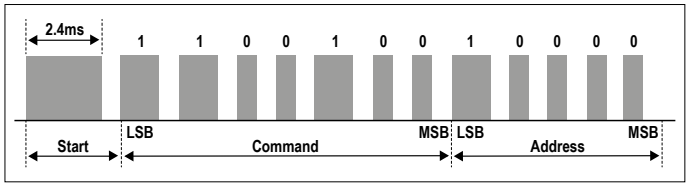


Bild 5. Beispiel für einen IR-Code bei SIRC von Sony.

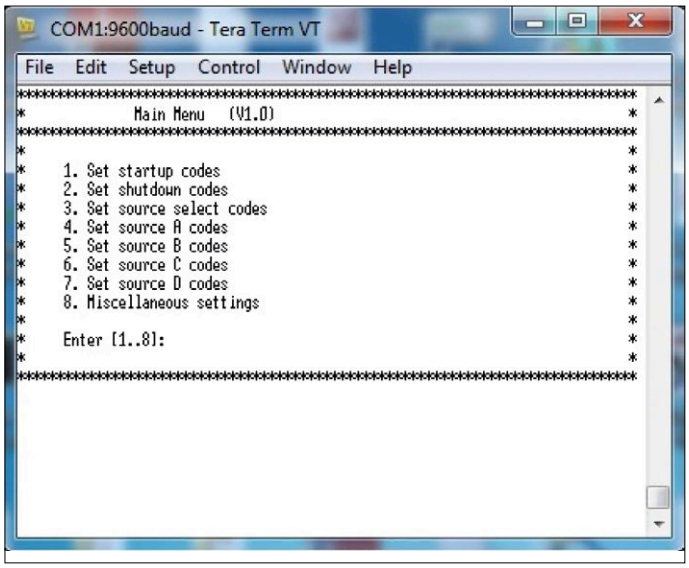


Bild 6. Im Fenster des Terminalprogramms erscheint ein Auswahlmenü.

koll Sirc von Sony und der Standard RC6 von Philips weisen fast identische Startimpulse auf, die Breite beträgt ungefähr 2,4 ms. Um sie voneinander unterscheiden zu können, wird auch die Pause bis zum zweiten Impuls gemessen. Beim Standard RC6 ist diese Pause 0,889 ms breit, bei Sirc 0,6 ms (siehe **Bild 4** und **Bild 5**).

Nachdem das Infrarot-Protokoll erkannt wurde, ist auch die Codierung der Daten-Bits bekannt. RC5 und RC6 arbeiten mit Biphasen-Codierung, NEC wendet eine Pulsdauer-Codierung an und Sony hat sich für die Pulsbreiten-Codierung entschlossen. Ausführliche Beschreibungen mit allen Details der betrachteten Infrarot-Standards findet man in den Quellen [2], [3], [4] und [5].

Der Mikrocontroller liest die Infrarot-Datenbits, packt sie zu Bytes und speichert sie in seinem EEPROM.
Beim Senden der Infrarot-Codes wird die Prozedur umgekehrt durchlaufen. Das erste Byte im EEPROM kennzeichnet das IR-Protokoll. Für die Software ist es die Information, wie die Datenbits codiert sind und welche Frequenz das Trägersignal haben muss. Anschließend werden die Datenbytes nacheinander in gleicher Folge gesendet, wie sie „gelernt" wurden. Beim Empfangen der IR-Codes hatte der IR-Empfänger (SFH5110-36) das Trägersignal unterdrückt, so dass es vor dem Senden erneut moduliert werden muss. Die Frequenz beträgt abhängig vom Protokoll 36 kHz, 38 kHz oder 40 kHz. Die originalen Philips- oder Sony-Fernbedienungen wiederholen den IR-Code so lange, wie der Taster gedrückt bleibt. Die Pausenzeit zwischen zwei wiederholten IR-Codes hängt vom Protokoll ab. Bei Philips ist ein Toggle-Bit eingefügt, das sich nicht ändert, solange der Taster gedrückt ist. Nach Loslassen des Tasters und erneutem Drücken wechselt das Toggle-Bit seinen Zustand.

NEC hat die Aufgabe anders gelöst. Hier wird der Kommando-IR-Code nur einmal gesendet, danach folgen Wiederhol-IR-Codes mit fester Zwischenpause, solange der Taster gedrückt ist. Manche Geräte erwarten mindestens einen Wie-

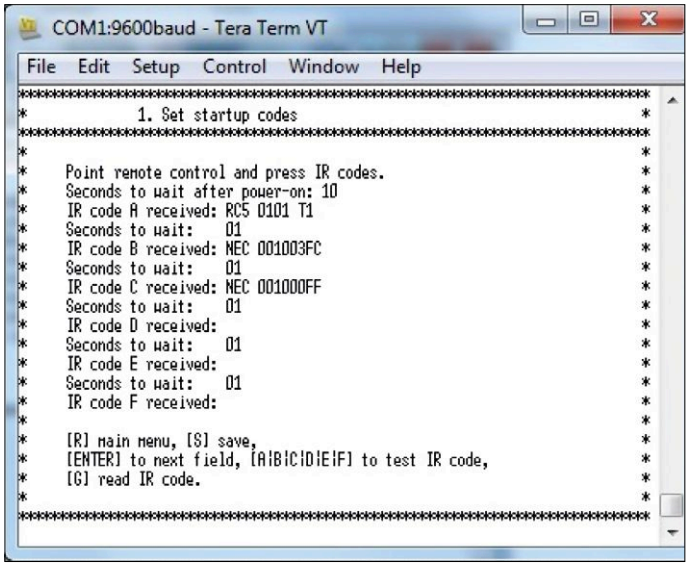


Bild 7. In Menü 1 werden die Einschaltcodes programmiert.

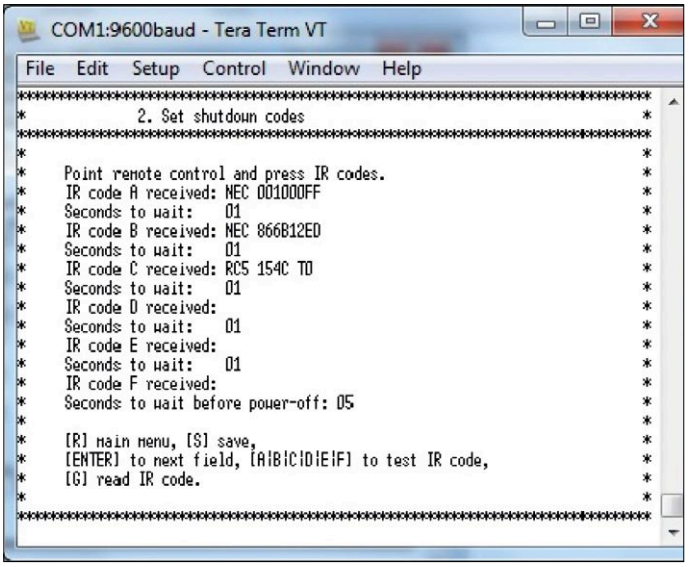


Bild 8. In Menü 2 werden die Ausschaltcodes programmiert.

derhol-IR-Code, bevor sie das Kommando ausführen. Deshalb hängt die Software immer einen Wiederhol-IR-Code an.

Das Senden der Infrarot-Signale lässt sich mit der Kamera eines Mobiltelefons überprüfen. Es wird in einigen Zentimetern Abstand vor der Infrarot-LED positioniert. Die Kamera ist auch für infrarotes Licht empfindlich. Die Infrarot-LED hat einen engen Abstrahlwinkel, sie muss deshalb möglichst senkrecht auf das Kameraobjektiv gerichtet werden.

IR-Codes „lernen“

Verbinden Sie die Schaltung über ein Nullmodem-Kabel mit dem RS232-Anschluss Ihres PCs (natürlich ist auch ein Adapterkabel RS232/USB verwendbar). Starten Sie auf dem PC ein Terminalprogramm (beispielsweise Tera Term) und stellen Sie die Parameter ein auf 9600 Baud, 8 Datenbit, ohne Parität, 1 Stoppbit. Wenn Sie die Betriebsspannung einschalten, erscheint im Terminalfenster ein Auswahlmenü (**Bild 6**). Der Infrarot-Empfänger muss sich im Schatten von Lichtquellen, Fernsehbildschirmen und Monitoren befinden, da das Licht den Empfang der Infrarot-Codes stört. Die Codes könnten dann fehlerhaft empfangen und gespeichert werden.

In Menü 1 (**Bild 7**) werden die IR-Codes für das Einschalten und in Menü 2 (**Bild 8**) für das Ausschalten programmiert. In Menü 3 (**Bild 9**) wird das fernzusteuernde Gerät gewählt. Die Menüs 4 bis 7 (**Bild 10**) dienen dazu, die IR-Codes der Geräte für die Taster Aufwärts, Abwärts, Links und Rechts zu programmieren. Abhängig vom gewählten Gerät sind hier für jeden Taster unterschiedliche IR-Codes programmierbar.
Beim Einlesen eines neuen IR-Codes muss über das Terminal „G“ gesendet werden (die Kommandos werden ausschließlich in Großbuchstaben geschrieben). Der IR-Code wird wiederholt eingelesen, indem das Kommando „G“ wiederholt wird, die empfangenen IR-Codes werden miteinander verglichen. Der am häufigsten empfangene IR-Code ist wahrscheinlich korrekt. Neue IR-Codes und Änderungen werden mit dem Kommando „S“ gespeichert. Beim Kommando „R“ kehrt das Terminal zum Hauptmenü zurück.

In Menü 8 können folgende Funktionen eingestellt werden:

1. Anzahl der Wiederholungen eines IR-Codes. Diese Option simuliert einen gedrückt gehaltenen Taster der Fernbedienung.

2. Einschalten der Funktion „Auto Power On“ des digitalen Eingangs.

In der Praxis

Über den digitalen Eingang lassen sich die ferngesteuerten Geräte ein- und ausschalten. Mit Jumper JP1 kann festgelegt werden, bei welchem logischen Signal an diesem Eingang dies geschieht.
Der Taster „Select“ ist jederzeit in Funktion, um die Start- und Stopp-IR-Codes zu senden. Um die Start-Codes zu senden, wird der Taster kurz gedrückt, so dass das Relais einschaltet. Anschließend werden nach programmierter Wartezeit die Start-IR-Codes gesendet.

Nachdem die Start-Codes gesendet wurden, kann unter den Geräten gewählt werden. Dazu wird der Taster „Select“ kurz gedrückt. Die IR-Codes der Richtungstaster Aufwärts, Abwärts, Links und Rechts variieren abhängig vom gewählten Gerät.
Um die Geräte auszuschalten, muss der Taster „Select“ mindestens zwei Sekunden gedrückt gehalten werden. Danach werden die Stopp-IR-Codes gesendet, zum Schluss fällt das Relais ab. ⏮

(160034)gd

Weblinks

[1] www.elektormagazine.de/160034
[2] www.sbprojects.com/knowledge/ir/nec.php
[3] www.sbprojects.com/knowledge/ir/rc5.php
[4] www.sbprojects.com/knowledge/ir/sirc.php
[5] www.sbprojects.com/knowledge/ir/rc6.php

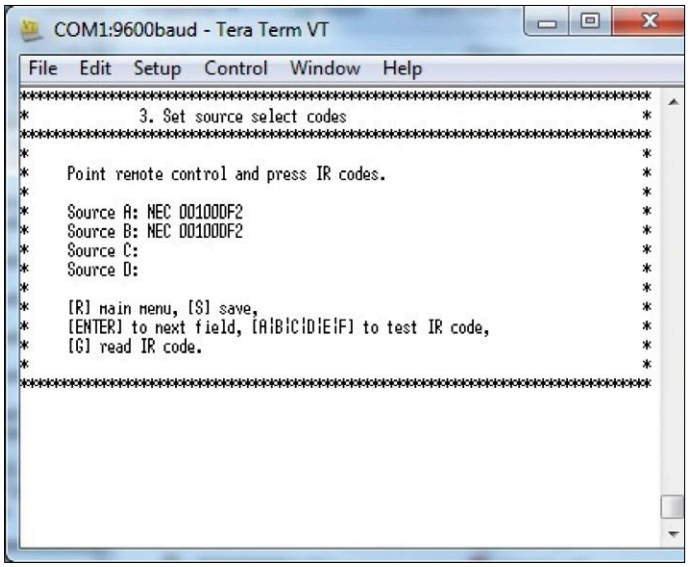


Bild 9. In Menü 3 wird das fernzusteuernde Gerät gewählt.

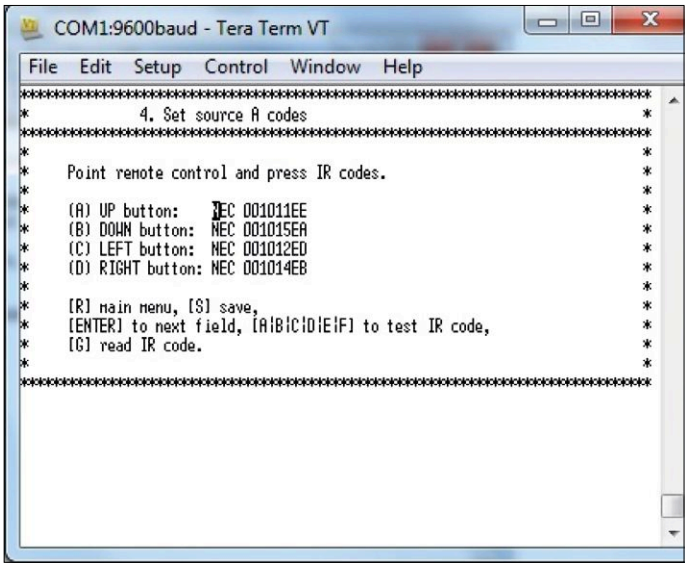


Bild 10. In Menü 4 bis 7 werden die Richtungstaster programmiert.

Ukule-LED

Ukulele lernen mit seriell adressierbaren RGB-LEDs

Von **Raghav Ayurkudi Subramaniam** (USA)

Ukule-LED ist eine Ukulele, in deren Griffbrett digital adressierbare RGB-LEDs eingebaut sind, die je nach zu spielendem Akkord in unterschiedlichen Konfigurationen aufleuchten können. Sie helfen dem Anfänger, die richtigen Akkorde auf Anhieb zu finden und können auch den Fortgeschrittenen bei komplexeren Griffen unterstützen.

Ukule-LED besitzt zwei Betriebsarten: Im Play-Modus kann der Anwender eine Song-Datei als Text in das System eingeben, in der das Tempo, die Schläge pro Takt sowie die Akkorde in der richtigen Reihenfolge enthalten sind. Die in den ersten vier Bünden des Griffbretts eingelassenen 16 LEDs leuchten beim Abspielen der Datei dann genau zum richtigen Zeitpunkt auf. Im Übungs-Modus kann der Anwender einen bestimmten Akkord auswählen, der durch die LEDs angezeigt wird. Die LEDs sind mit einem Atmel-Mikrocontroller verbunden, der an der Ukulele befestigt ist. Das Instrument kann an einen PC angeschlossen werden, auf dem ein zugehöriges Programm mit Kommandozeilen-Interface aktiv ist.

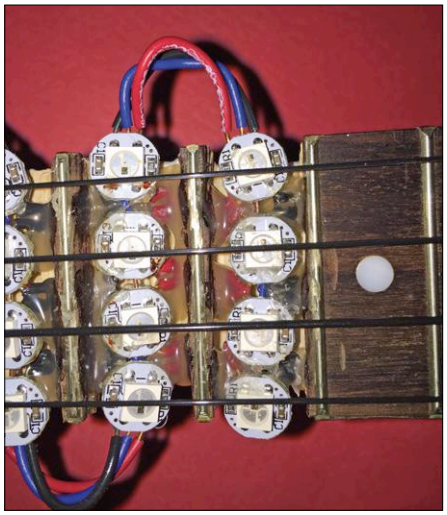


Bild 1. Fertiges Ukule-LED-Griffbrett aus der Nähe betrachtet. Die 16 RGB-LEDs sind seriell miteinander verbunden und an einen einzigen Mikrocontoller-I/O-Pin angeschlossen.

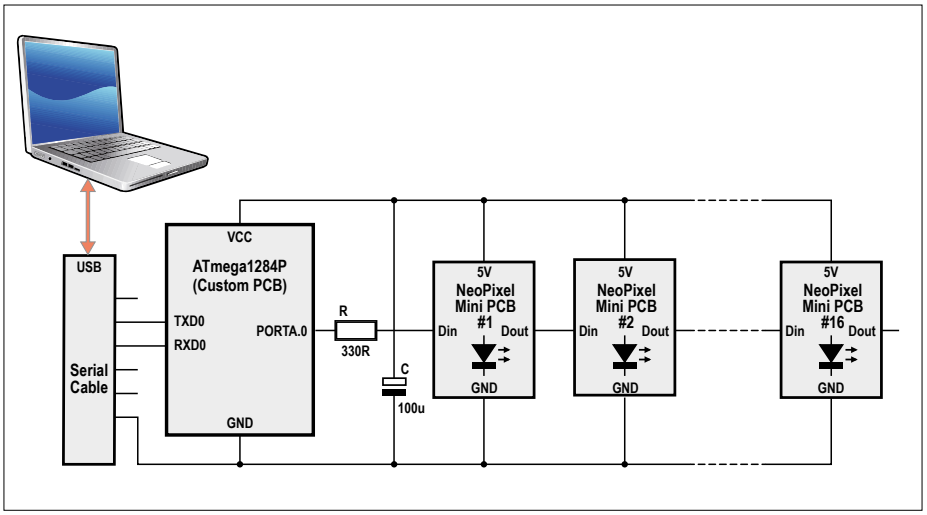


Bild 2. Blockschaltbild der gesamten Hardware einschließlich Mikrocontroller, RGB-LEDs und serieller Verbindung zum PC.

Eigenschaften

- RGB-LED-betriebenes System zum Erlernen eines Musikinstrumentes
- Spiel- und Übungsmodi
- Leicht verständliches Dateiformat

Hardware

Wir haben uns für den Mikrocontroller ATmega1284P von Atmel entschieden, da dieser zwei für das Projekt benötigte 16-Bit-Timer enthält. Einer dient zur zeitlichen Koordinierung der Tasks und einer wird für musikalische Zwecke genutzt. Zusätzlich haben wir die Standardplatine von Bruce Land für den ATmega1284P [2] verwendet, die unter anderem einen externen Quarz, eine LED und die Stromversorgung enthält und die Verbindung zum Mikrocontroller sehr einfach gestaltet. Das Elektor-Platino-Board kann ebenfalls zusammen mit einem ATmega1284P verwendet werden.

Zum Einbau der LEDs ins Griffbrett haben wir eine Fräse von Dremel verwendet. Beim Fräsen muss sehr genau darauf geachtet werden, dass Griffbrett und Bünde nicht beschädigt werden und die LEDs nicht aus dem Griffbrett herausragen, damit die Saiten nicht durch die LEDs am Schwingen gehindert werden oder diese selbst als Bünde fungieren. Zur Befestigung der LEDs haben wir eine Heißklebepistole benutzt. **Bild 1** zeigt das fertig bearbeitete Griffbrett der Ukulele. Für die LEDs wurden „NeoPixel"-Platinen von Adafruit verwendet, von denen jede eine seriell ansteuerbare RGB-LED des Typs WS2812B enthält. Dadurch lassen sich alle LEDs von einem einzigen Mikrocontroller-I/O-Pin aus über ein im Timing sehr empfindliches 1-Draht-Protokoll kontrollieren. Die Masse sowie die Ausgänge für die Daten und die Versorgung der ersten Miniplatine werden dabei mit den entsprechenden Eingängen bzw. dem Masse-Anschluss der zweiten Platine verbunden, und so weiter. Der Daten-Eingangs-Pin der ersten Platine ist mit dem betreffenden I/O-Pin des Mikrocontrollers über einen Widerstand von 330 Ω verbunden, um die erste LED vor eventuellen Spannungsspitzen zu schützen. Zusätzlich wurde ein Kondensator von 100 µF zwischen Versorgung und Masse geschaltet, damit die erste LED nicht durch hohe Einschaltströme gefährdet ist (gemäß *NeoPixel PCB Documentation* von Adafruit [3]).

Das individuelle Verlöten der 16 einzelnen LEDs war die zeitaufwändigste Aktion beim Umbau, da zwischen den einzelnen LED-Platinen innerhalb eines Bundes nur ein Abstand von einem bis zwei Millimetern war. Die auf den Bünden sitzenden, vier Gruppen von LED-Platinen konnten dann mit etwas längeren Drähten verbunden werden. Zum Schluss wurde dann die Einheit von 16 NeoPixel-Platinen über drei lange Kabel an die Hauptplatine angeschlossen.

Nun galt es, die LEDs optimal mit Strom zu versorgen. Da zur gleichen Zeit nur vier LEDs leuchten, deren Intensität nicht maximal sein musste, konnten wir auf eine eigene Versorgungsquelle für die LED-Platinen verzichten und diese einfach mit dem VCC-Anschluss des Mikrocontrollers verbinden. Um das gesamte

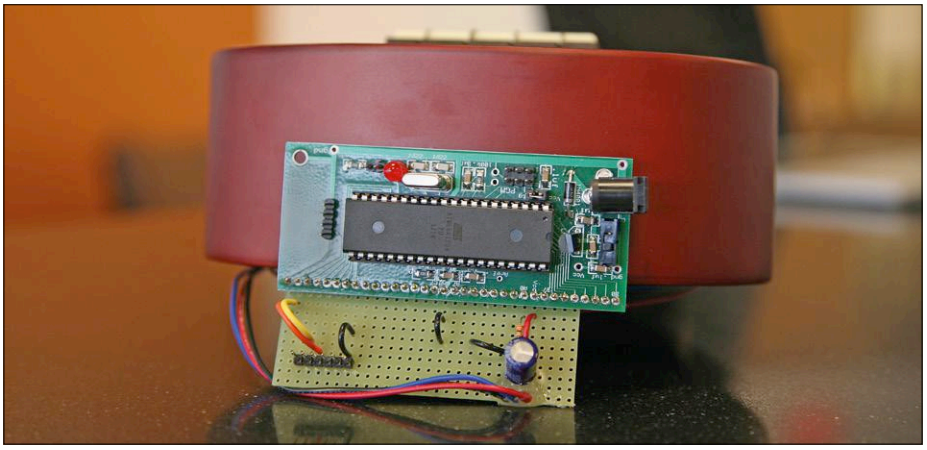


Bild 3. Die Platinenanordnung wurde am Ende der Ukulele festgeklebt. Die Lochrasterplatine enthält alle zusätzlichen Bauteile zur Ansteuerung der LEDs und zur seriellen Kommunikation.

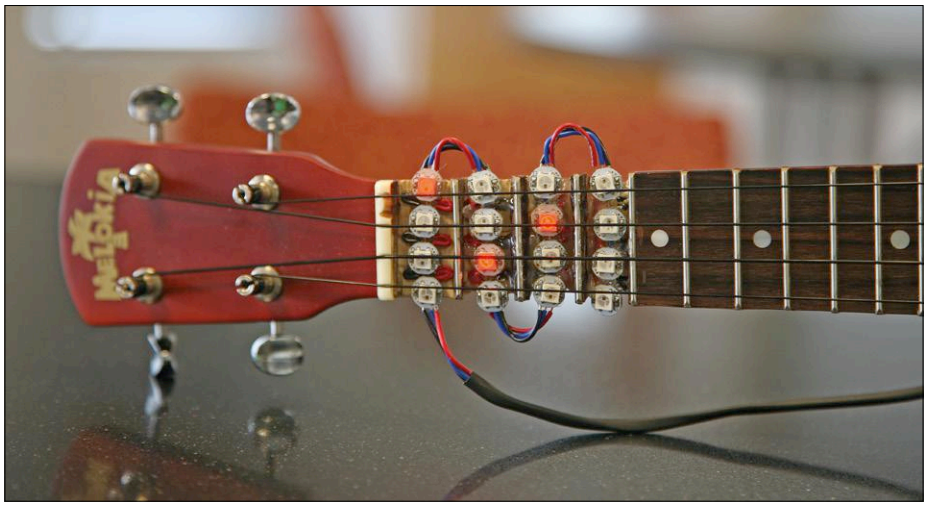


Bild 4. G-Moll-Akkord. Bei Moll-Akkorden leuchten die LEDs rot.

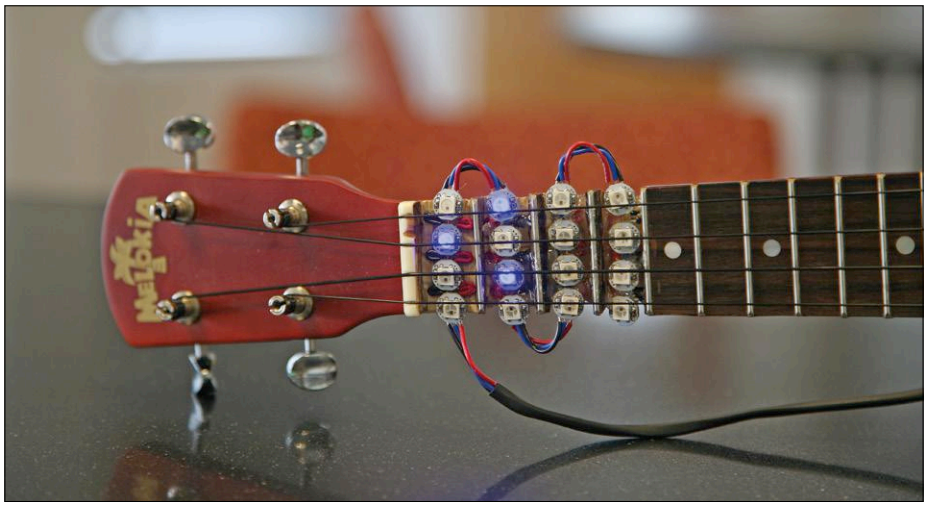


Bild 5. G7-Akkord. Bei Septim-Akkorden leuchten die LEDs blau.

System zu versorgen, reicht ein einziges 9-V-Netzteil vollkommen aus. **Bild 2** zeigt ein Blockschaltbild der in der Ukulele verwendeten Elektronik. Zum Schluss montierten wir alle elektronischen Einheiten wie zum Beispiel Mikrocontroller, Kondensator und Anschlüsse für die erforderlichen Kabel auf einer kleinen Basisplatine, die wir an der Rückseite der Ukulele festklebten (**Bild 3**).

Kommandozeilen-Interface

Zur Zusammenarbeit mit dem Mikrocontroller benötigt unser Projekt einen Computer, auf dem ein auf Python basierendes Kommandozeilen-Interface läuft (CLI). Wir verwendeten dazu *docopt* [4], einen Kommandozeilen Argument-Parser für Python. Es gibt zwei Befehlsformate, mit denen der Anwender eine Ukule-LED-Session starten kann:

```
ukule_led practice „chord“ port „portname“
ukule_led play „file“ port „portname“
```

Der erste Befehlstyp weist den Mikrocontroller an, in den Übungsmodus überzugehen. Die Argumente für diesen Modus sind `chord`, was dem zu spielenden Akkord C, D, Fis usw. entspricht. Das andere Argument lautet `portname`, was dem Namen des seriellen Ports entspricht, mit dem der PC über USB mit dem Mikrocontroller verbunden ist.
Der zweite Befehlstyp dient zum Übergang in den Play-Mode. Die Argumente lauten `file`, ein Verweis auf den Pfad, in welchem die Song-Datei auf dem Computer gespeichert ist, und `portname` (siehe oben). Das Dateiformat der Song-Dateien für die Ukule-LED haben wir selbst entwickelt, da die existierenden Standards wie zum Beispiel MIDI zu umständlich gewesen wären. Hier ein Beispiel einer Song-Datei:

```
240 4
|A B C D|E F G -|A - - B|
|A - Bbm A7|Bm X A#7 -|A - X D|
```

Die erste Zeile der Song-Datei enthält zwei Zahlen: Das Song-Tempo in *beats per minute*, und dessen Zeitsignatur in *beats per measure* (Schläge pro Takt). Der Rest der Datei enthält die einzelnen Takte des Stückes mit den zu spielenden Noten, die durch das Zeichen „|“ getrennt werden. Pro Taktschlag kann ein Akkord dargestellt werden. Unterstützt werden Dur-, Moll- sowie Major-, Minor- und Septim-Akkorde, wobei jedem Akkord-Typ eine bestimmte Farbe zugeordnet wird: Major = grün, Minor = rot, Septim = blau. Die **Bilder 4 und 5** zeigen dazu einige Beispiele. Ein Akkord kann auf jedem Halbton der Tonleiter basieren. Ein Bindestrich bedeutet, dass ein Akkord auch für den nächsten Taktschlag beibehalten wird, und ein X bedeutet, dass während eines bestimmten Taktschlags kein Akkord gespielt werden soll.

Wenn das Python-Interface (CLI) einen Befehl von Anwender empfängt, werden die Befehlsargumente mit docopt verarbeitet. Der Befehl wird daraufhin in einen String konvertiert und seriell weiterverarbeitet. Zu spielende Akkorde werden mit einer „1“ markiert und von 1 an durchnummeriert. Erscheint ein X (Akkord wird nicht gespielt), so wird die Zahl 0 eingesetzt. Ein Übungsstring sähe zum Beispiel wie folgt aus: „1/17“, und ein Play-String könnte folgende Struktur besitzen: “2/200/4/14/13/02/30/-1”. Die erste Zahl im String hat folgende Bedeutung: „1“ bedeutet Übungsmodus, und „2“ heißt „Play-Modus“. Im Übungsmodus steht die nach der „1“ folgende Zahl für den zu spielenden Akkord. Im Play-Modus gibt die auf die „2“ folgende Zahl das Tempo an, und die dritte Zahl steht für die Anzahl der Schläge pro Takt; im Beispiel 200 und 4. Die dann folgenden Zahlen repräsentieren die Akkorde in einem Stück in zeitlicher Reihenfolge. Ihre Anzahl ist unbegrenzt. Das Ende der Liste wird mit „-1“ markiert. Mit Hilfe des *pySerial package* von Chris Liechti [5]

öffnet das Programm eine Verbindung zu einem seriellen Port, zu welchem der String dann gesendet wird.

Firmware

Der Mikrocontroller führt zwei Hauptaufgaben (Tasks) aus. Erstens: Empfang der Befehle vom CLI über einen seriellen Port. Der Mikrocontroller besitzt in seinem Speicher ein Akkord-Array, bei dessen Indizes es sich um Akkord-Nummern handelt. Unter den entsprechenden Index-Zahlen sind die Konfigurationen der zu den Akkordnummern gehörenden LEDs gespeichert.
Zweitens: Jede hundertstel Sekunde wird das Akkord-Array mit dem eingestellten Tempo über einen Pointer abgefragt und die LED-Konfiguration aktualisiert. Im Übungsmodus bleibt dieser Pointer konstant (**Bild 6**).
Für das richtige Timing der beiden Tasks haben wir den von Dan Henriksson and Anton Cervin entwickelten *TinyRealTime (TRT)* Kernel [6] verwendet – dazu noch den *WS2812 LED driver* von Alan Burlison [7]. Dieser Treiber ist in Assembler geschrieben und kümmert sich um die zeitliche Steuerung und Farbgebung der Leuchtdioden.

Der Code enthält eine Hauptfunktion, die zuerst den LED-Treiber aktiviert, indem sie den I/O-Port angibt, mit dem die LEDs physikalisch verbunden sind. Als nächstes wird der TRT-Kernel gestartet. Er verwendet Timer 1, einen der beiden 16-bit-Timer des ATmega. Die main-Schleife startet anschließend den nächsten 16-Bit-Timer (Timer 3), der für die zeitlichen Übergänge zwischen den Akkorden sorgt.
Der von der Funktion `serialComm` ausgeführte, erste Realtime-Task wartet auf eine zu verarbeitende Eingabe des Anwenders am seriellen Port. Zusätzlich wird ein Command-Buffer (cmd) von 1000 Bytes freigegeben; groß genug, um eine große Zahl von Songs zu speichern. Danach wird die TRT-Funktion `gets(cmd)` aufgerufen, die auf eine Datenstruktur (Semaphor) wartet, die ausgegeben wird, wenn TRT eine Eingabe vom seriellen Port erhält. Wenn die Funktion den Wartevorgang beendet, ist ein Befehl in cmd gespeichert. Der Befehl besteht aus einer Gruppe von aus dem String extrahierten Zahlen, die seriell vom CLI aus gesendet werden.
Bei einem Übungsmode-Befehl setzt `serialComm` die Variable `chord_idx` auf die Nummer des einzelnen, anzuzeigenden Akkords. `chord_idx` ist der ins Akkord-Array zeigende Pointer. Der Wert von `chord_idx` bleibt bis zum Empfang eines weiteren Befehls konstant.
Im Play-Modus liest `serialComm` die zweite Zahl innerhalb des Befehls: Das Tempo der *beats per minute* im Song. Der Wert wird nach folgender Formel in eine Periode des Timers 3 konvertiert:

Periode = (62500 × 60) / Tempo − 1

Nachdem die Periode des dritten Timers gesetzt ist, extrahiert die Funktion die dritte Zahl aus dem Befehl: Die Anzahl der Schläge pro Takt. Schließlich ruft `serialComm` eine Hilfsfunktion auf, die das cmd-Array durchsucht, die als Zeichen codierten Zahlen in Integer verwandelt und sie in ein globales Array mit der Bezeichnung song einsetzt. Mit dem Timer 3 ISR wird song dann „abgespielt“. Jedes Mal, wenn ISR „feuert“, wird `chord_idx = song[song_idx]` gestartet. Der aktuelle Index von Song, song_idx, wird anschließend inkrementiert. Der zweite Real-Time-Task fragt den vom ISR gesetzten Wert von `chord_idx` zyklisch ab, erhält die entsprechenden Daten aus `lookup_table` und gibt sie an den LED-Treiber weiter.

Danksagung

Ich danke Dr. Bruce Land und meiner technischen Assistentin, Eileen Liu, für ihre Unterstützung. Das Projekt Ukule-LED war auf Grund der von folgenden Autoren geschriebenen, Open-Source-Software problemlos realisierbar: Dan Henriksson, Anton Cervin, Jeorg Wunsch, Alan Burlison, Chris Liechti und die Entwickler hinter docopt. Letztendlich wäre das Projekt auch ohne meinen Labor-Partner, Jeff Tian, nicht möglich gewesen.

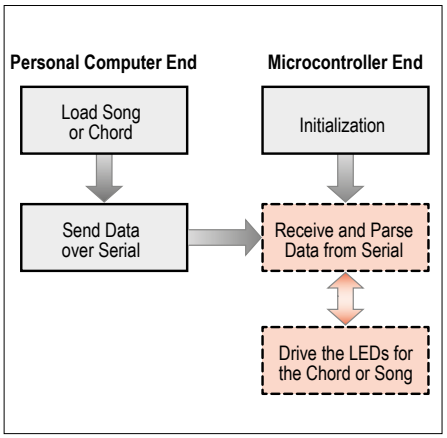


Bild 6. Dieses Flussdiagramm zeigt die verschiedenen Software-Tasks der Schaltung: Zwei Real-Time-Tasks im Mikrocontroller und ein Kommandozeilen-Interface, das auf dem PC läuft.

Zusammenfassung

Mein Ziel bei diesem Projekt bestand in der Entwicklung eines erweiterbaren Systems zum Erlernen des Ukulele-Spielens, wobei das Instrument auch nach Einbau der Elektronik leicht und handlich bleibt. Das Software-Interface ist dank docopt einfach anzuwenden. Ukule-LED kann durch Editieren der Dateien `chord_map.py` und `chord_lookup_table.h` leicht für alle möglichen Akkorde erweitert werden. In den erwähnten Dateien sind die unterstützten Akkorde und die Konfigurationen und Farben gespeichert. ◀

(160439)

Weblinks

[1] www.elektormagazine.de/160439
[2] http://people.ece.cornell.edu/land/PROJECTS/ProtoBoard476/
[3] https://learn.adafruit.com/adafruit-neopixel-uberguide
[4] http://docopt.org/
[5] http://pyserial.sourceforge.net/
[6] http://www.control.lth.se/~anton/tinyrealtime/
[7] https://sourceforge.net/projects/abavr/

Vielsagende Positionslichter

Für alle Flugmodelle

Ein Flugmodell sollte seinem großen Vorbild auch im Detail folgen. Die Beleuchtungseinrichtungen machen da keine Ausnahme. Doch das soll uns nicht daran hindern, diese Lichter mit einer kleinen Zusatzaufgabe auszurüsten: Sie übermitteln hier auch Infos zum Akkufüllstand des Modells.

Von **Udo Hunzelmann**, DK8VT@DARC.de

In der Anlage 1 der LuftVO sind die Vorschriften über die von Luftfahrzeugen zu führenden Lichter aufgeführt [1]. In §2 ist da zu lesen, dass ein Flugzeug ein rotes Licht links und ein grünes Licht rechts führen muss, das in Flugrichtung gesehen links beziehungsweise rechts über einen Winkel von 110 Grad nach oben und unten scheint. Dazu kommt ein weißes Licht, das ungehindert nach hinten rechts und links über einen Winkel von 70 Grad nach oben und unten scheint, also am besten am Heck angebracht ist. Die Lichter dürfen dauerhaft leuchten oder Blinklichter sein.

Neben den Positionslichtern schreibt der §3 blinkende Zusammenstoß-Warnlichter (Anti Collision Light, ACL) vor, die möglichst aus allen Richtungen zwischen 30° über und 30° unter der Horizontalebene des Luftfahrzeugs zu sehen sind. Sind Luftfahrzeuge mit ACLs ausgestattet, müssen die Positionslichter dauerhaft leuchten.

Blitzlicht mit Zusatzinformation

Was das Gesetz für echte Flugzeuge (und Hubschrauber) vorschreibt, sollte einem Modellflugzeug billig sein. Doch da wir es als Modellpiloten mit den gesetzlichen Vorschriften für die Großen nicht ganz so genau nehmen müssen, steht es uns

frei, den Positions-/AC-Lichtern/Blitzern neben ihrer originären noch eine kleine, aber überaus nützliche Zusatzfunktion zukommen zu lassen: eine weithin sichtbare Akku-Füllstandsanzeige!

Bei der Modellfliegerei besteht nämlich immer das Problem, dass man keinen Schimmer hat, wie voll oder erschöpft die Batterien im Modell sind. Um diesem Problem abzuhelfen, habe ich eine kleine Blitzerschaltung entwickelt, die das Blitzverhalten je nach Akkuladung beeinflusst. Zusätzlich kann ein Summer mit der Schaltung gekoppelt werden, der den Standort des Modells akustisch vermeldet.

Die Schaltung in **Bild 1**, die am Servo-Ausgang des Flugcontrollers im Modell über K1 angeschlossen wird, zeigt einen kleinen ATtiny45-Mikrocontroller mit nur acht Füßchen, der drei Schaltausgänge mit N-Kanal-MOSFETs steuert. Während der Summer an K2 nur wenig Strom zieht (bei der geringen Betriebsspannung in der Größenordnung von 5 mA, abhängig vom Modell), so dass ein Kleinsignal-Schalttransistor BS170 völlig ausreicht, haben die MOSFETs für die beiden Hochleistungs-LEDs der Blitzer an K3 und K4 ein schon etwas größeres Kaliber. Die IRF540 im TO220-Gehäuse werden auch mit den hohen Strömen durch die LEDs spielend fertig. Die Teilschaltungen mit den Shunts R3 und R6 und den Bipolartransistoren BC547C stellen einfache Strombegrenzungen dar und sorgen dafür, dass die MOSFET-Stufen im Falle eines Kurzschlusses (der LEDs) sofort deaktiviert werden.

Die Schaltung wird aus dem Akkupack (mit mindestens zwei LiPo-Zellen) über dessen Balancer-Anschlusskabel [2] versorgt. Auf der kleinen Platine in **Bild 2** stehen dafür drei gewöhnliche Stiftleisten-Pins (K5) zur Verfügung. Während die Hochleistungs-LEDs die volle Spannung (VCC) von 7,4 V erhalten, kommen der Controller und der Summer mit der Hälfte (VCC/2), also 3,7 V aus.

Der ATtiny45 hat eine wunderbare interne Funktion: Er kann die Spannung an seinem VCC-Pin analog messen! Dies habe ich natürlich ausgenutzt und ohne ein einziges zusätzliches Bauteil (sondern nur per Software) eine Art Fensterkomparator für VCC/2 aufgebaut:

- **VCC/2 > 3,7 V**
  Akku ist voll: 3x blitzen
- **3,4 V ≤ VCC/2 ≤ 3,7 V**
  Akku ist halbvoll: 2x blitzen
- **VCC/2 <3,4 V**
  Akku ist leer: 1x blitzen

Einen kleinen Haken hat dieser Aufbau aber dann doch, es wird nämlich immer nur die Spannung der ersten Zelle stellvertretend für alle anderen gemessen. Erst in einer zukünftigen erweiterten Schaltung sollen die gesamte Akkuspannung

INFOS ZUM PROJEKT

Modellflug
Positionsbeleuchtung
ACL

➡ Einsteiger
Fortgeschrittener
Experte

etwa eine Stunde

Lötkolben, ATtiny45-Programmiergerät, Arduino-IDE

rund 20 Euro

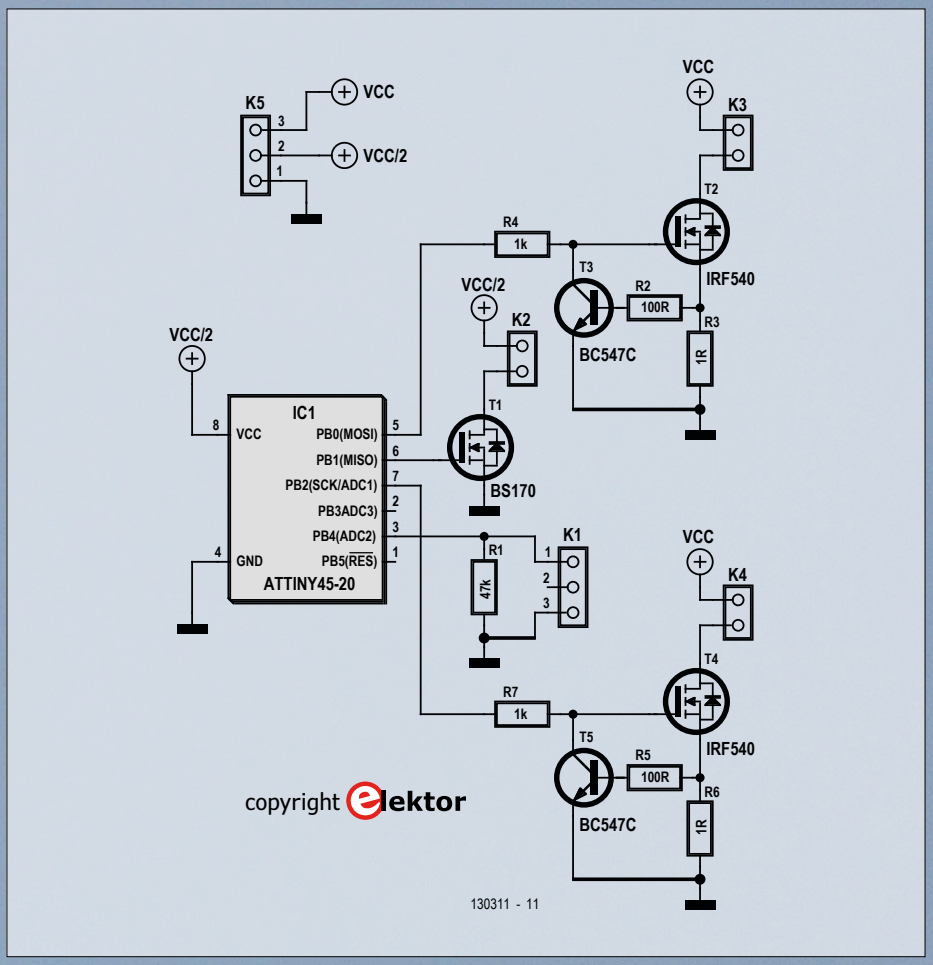


Bild 1. Schaltung des Modellflug-Blitzers.

und damit alle Zellen gemessen werden. Der Aufbau der Schaltung auf der Platine, die im Elektor-Shop erhältlich ist, deren Layout aber auch von der Projektseite [3] heruntergeladen werden kann, ist in einer knappen Stunde erledigt. Besondere Schwierigkeiten sind nicht zu erwarten, zumal nur Durchsteckbauteile und keine SMDs verwendet werden.

Software in drei Modi

Durch die Software lässt sich die Blitzer-Schaltung in drei Modi betreiben, die sich von einem 3-Wege-Schalter der Fernsteuerung auch während des Flugs oder auch nach der Landung umschalten lassen.

1. Schalter hinten
Blitzer und Summer aus.

2. Schalter Mitte
Blitzer ein und Summer aus.

3. Schalter vorne
Blitzer ein und Summer ein.

Die Mittelstellung ist für den Flug gedacht, die dritte für danach. Sollte das Modell crashen (was keinem zu wünschen ist, aber es sind auch schon Meister vom Himmel gefallen...), kann man es durch den Summton auch im unwegsamen/unübersichtlichen Gelände leicht wiederfinden und bergen. Der Summton ist in der Software auf gut 4 kHz eingestellt, Sie können die Tonhöhe aber leicht ändern, so wie es Ihnen am besten gefällt.
Die Software [3] wurde zunächst in BASCOM entworfen und später in der Arduino-IDE auf C portiert. Die Entscheidung, welcher Modus aktiv ist, wird direkt zu Beginn in der Hauptschleife getroffen und hängt von der Länge des Servoimpulses (kleiner 1,25 ms, 1,25...1,75 ms und größer 1,75 ms) ab. Im zweiten Modus, wenn das Blinken der LEDs Aufschluss über die Restkapazität der Akkus geben soll, wird die Akku-Versorgungsspannung des ATtiny45 gemessen und ausgewertet. Wie einfach das funktioniert, kann man dem Auszug aus dem Programm in **Listing 1** entnehmen.

Akkuzustand im Blick

Im vergangenen Herbst habe ich zwei Exemplare des Blitzers in meine beiden Elektrosegler eingebaut. Dabei habe ich feststellen können, dass man die Blitz-LEDs bei hellem Sonnenschein, guter

Listing 1. VCC-Messung durch den ATtiny45 (Programmauszug).

```
void measureBattery() {
  batteryVoltage = readVcc();
  if (batteryVoltage >= 3800) { //more than 3.7V: battery full
    ledBlink(3);
  }
  if (batteryVoltage >= 3500 && batteryVoltage < 3800) {
  //between 3.3V and 3.7V, apparently there is a small ofset
    ledBlink(2);
  }
  if (batteryVoltage < 3500) { //less than 3.4V: battery empty
    ledBlink(1);
  }
}

long readVcc() {
  // Read 1.1V reference against AVcc
  // first set the reference to Vcc and the measurement to the internal 1.1V reference
    ...
   #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
     ADMUX = _BV(MUX3) | _BV(MUX2);
   ...
   delay(2); // Wait for Vref to settle

  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA, ADSC)); // measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  long result = (high << 8) | low;

  result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  return result; // Vcc in millivolts
}
```

STÜCKLISTE

Widerstände:
R1 = 47 k
R2,R5 = 100 Ω
R3,R6 = 1 Ω
R4,R7 = 1 k

Halbleiter:
T1 = BS170
T2,T4 = IRF540
T3,T5 = BC547C
IC1 = ATtiny45-20PU (DIP 8), 130111-41

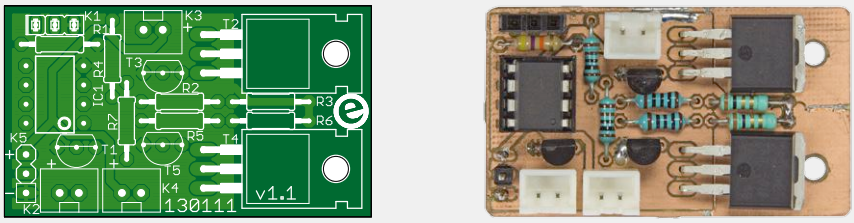


Bild 2. Platinenbestückung leichtgemacht: SMDs sind nicht vorhanden.

Außerdem:
K1 = 3-polige Buchsenleiste, Raster 0,1"
K2,K3,K4 = Ummantelte XH-Stiftleiste (JST bei RS Components 820-1554) plus passendes Buchsengehäuse (RS Components 820-1611) mit Crimpkontakten (RS Components 820-1529)
K5 = 3-polige Stiftleiste, Raster 0,1"
Platine 130111-1

Sicht und einer geschätzten Flughöhe des Modells von 100 m gut erkennen konnte, auch wenn das Modell beinahe einen halben Kilometer entfernt war. Ich hatte dabei gewiss keine übersinnliche Erscheinung, denn dies wurde mir auch von anderen Modellflugpiloten bestätigt. Der testweise im Flug eingeschaltete Summer konnte in einer Flughöhe von ungefähr 30 m bis in eine Entfernung von 50...60 m gehört werden. Aber das ist natürlich auch abhängig davon, wo man den Summer am Modell anbringt und um welchen Typ es sich handelt. Es gibt Summer, die bei dieser niedrigen Betriebsspannung von gut 3 V mit 72 dB schüchtern herumflüstern und solche, die unter gleichen Bedingungen einen lautstarken Alarmton von mehr als 105 dB erzeugen! Hier kann also durch eine geschickte Auswahl die akustische Reichweite optimiert werden.

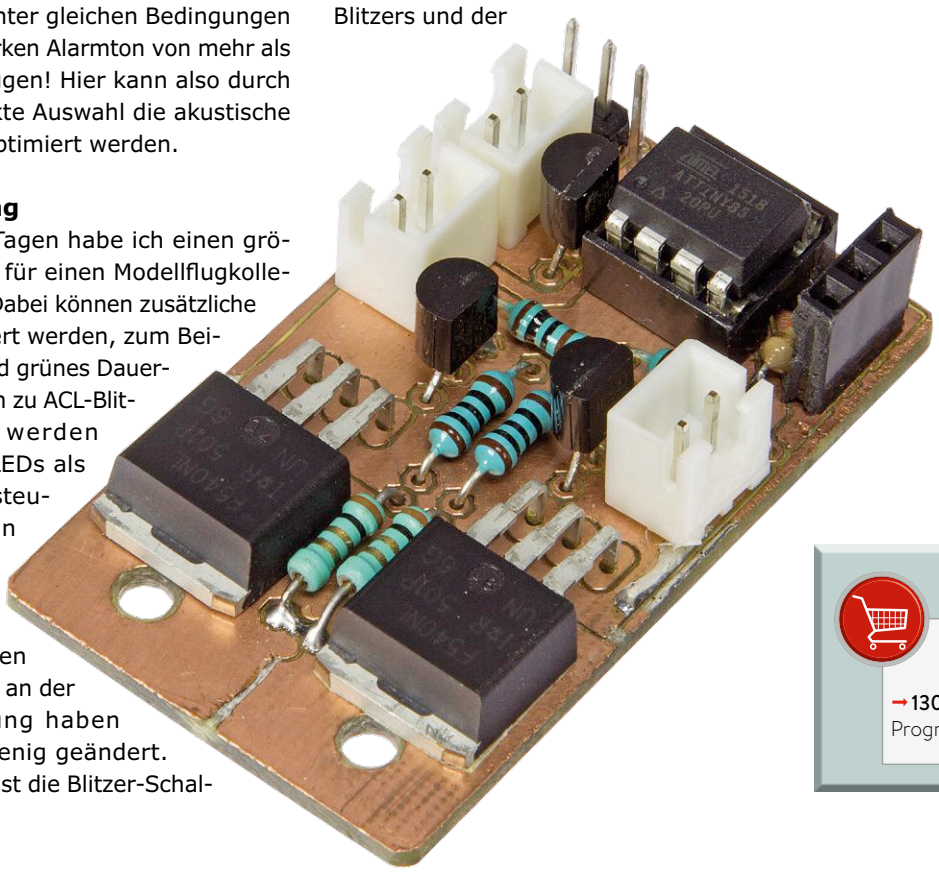
Erweiterung

Vor einigen Tagen habe ich einen größeren Blitzer für einen Modellflugkollegen gebaut. Dabei können zusätzliche LEDs gesteuert werden, zum Beispiel rotes und grünes Dauerlicht zusätzlich zu ACL-Blitzern. Auch werden zwei weiße LEDs als Blitzer angesteuert und ein Summer ist auch vorhanden. Die Funktionen des Schalters an der Fernsteuerung haben sich dabei wenig geändert. In Position 1 ist die Blitzer-Schaltung deaktiviert, in Mittelstellung arbeiten alle LEDs wie vorgeschrieben (und der Summer ist abgeschaltet) und in der Position 3 sind die blitzenden LEDs ab- und der Summer eingeschaltet.

Gute Aussichten

Zur Zeit baut ein Modellflugkollege diesen erweiterten Blitzer in seinen Elektrosegler mit einer Spannweite von 4,5 m ein. Bald kann der erste Testflug mit dem Modell und dem Blitzer erfolgen. Im Moment beschäftige ich mich mit einer Erweiterung des Blitzers auf sechs Funktionen. Dazu nutzt der Blitzer zwei Servoanschlüsse des Empfängers. Am Sender kann dann zur Steuerung des Blitzers und der weiteren Funktionen ein 3-Wege-Schalter und ein Schieberegler benutzt werden. Es würde mich freuen, wenn ich Rückmeldungen dazu bekäme, ob die Blitzer in ihren verschiedenen Ausführungen nachgebaut werden. Auch weitere Anregungen und Verbesserungen wären mir höchst willkommen. ◀

(130111)



Weblinks

[1] www.buzer.de/s1.htm?g=luftvoanl_1&a=1-6

[2] http://cache.freescale.com/files/32bit/doc/app_note/AN4428.pdf

[3] www.elektormagazine.de/130111

EINKAUFSZETTEL

→ 130111-1
Leerplatine

→ 130111-41
Programmierter Controller

ELEKTOR ETHICS

Smarte Geräte!

Was holen wir uns da ins Haus?

Von **Tessel Renzenbrink** (NL)

> Die Tür weigerte sich, zu öffnen. Sie sagte: „Fünf Cent, bitte."
> Er suchte in seinen Taschen. Keine Münzen mehr, nichts. „Ich werde dich morgen bezahlen", sagte er zur Tür. Noch einmal drehte er am Türknauf. Wieder blieb sie fest verschlossen.
> „Was ich dir bezahle", sagte er, „ist nur ein Trinkgeld. Ich muss dich nicht bezahlen."
> „Das sehe ich anders", sagte die Tür. „Schau in den Kaufvertrag, den du unterschrieben hast." In seiner Schreibtischschublade lag der Vertrag. Seit er ihn unterschrieben hatte, hielt er es schon manchmal für nötig, sich auf das Dokument zu berufen. Aber dort stand es: Zahlungen an die Tür für das Öffnen und Schließen sind eine obligatorische Gebühr, kein Trinkgeld.
> „Du siehst, dass ich recht habe", sagte die Tür. Es klang recht selbstgefällig.
> Aus der Schublade neben der Spüle nahm er ein Küchenmesser. Mit ihm begann er systematisch, die Verschraubung der geschäftstüchtigen Tür zu demontieren.
> „Ich werde dich verklagen", sagte die Tür, als die erste Schraube zu Boden fiel.
>
> Philip K. Dick, Ubik

Was holen wir uns ins Haus, wenn wir smarte Geräte kaufen? Wen lassen wir hinein, wenn wir unsere alten, einfältigen Geräte gegen intelligente, mit dem IoT vernetzte Objekte tauschen? In dem Science-Fiction-Roman *Ubik* von Philip K. Dick ist der Protagonist unaufhörlich mit intelligenten Objekten konfrontiert, die erbarmungslos einen Obolus verlangen, bevor sie ihm zu Diensten sind. Als der Roman 1969 erschien, war dieses Zukunftsbild ein Anlass zum Schmunzeln. Rückblickend aus dem Jahr 2017 war das Ganze schon damals visionär.

Resoluter Hersteller

Eine reale Manifestation von Philip K. Dick's Roman ist der Konflikt, der Anfang dieses Jahres zwischen einem Käufer des *Garadged* und dessen Hersteller entbrannte. Garadged ist ein IoT-Objekt, das über eine Cloud ein Garagentor bedient. Dem Käufer, der sich *R. Martin* nannte, gelang es nicht, das Objekt zu installieren. Er bat die Garadged-Community auf dem Herstellerforum um Hilfe, doch vergeblich. Wutentbrannt postete er auf Amazon eine vernichtende Produktbewertung.
Das wollte der Hersteller des Garadged nicht auf sich sitzen lassen. Er schlug zurück, indem er das Exemplar kurzerhand deaktivierte. Für ihn war das nicht schwierig, denn die Steuerung lief über eine Cloud-Plattform, die er administriert. Auf dem Forum teilte er lapidar mit, dass für den Garadged mit der ID 2f0036... der Zugang zum Server gesperrt sei. Der Käufer könne die Ware nur noch an Amazon zurückschicken, damit Amazon den Kaufpreis erstattet.

Vernetzte Beziehungen

Mit dem Einzug intelligenter Geräte ändert sich grundlegend das Verhältnis zwischen Konsument und Hersteller. Bisher war der Kauf eines Produkts eine einmalige Transaktion: Wir kaufen einen Kühlschrank, lassen ihn liefern und stecken den Stecker in die Netzsteckdose. Während der nächsten zehn Jahre müssen wir uns nicht um ihn kümmern, und mit dem Verkäufer oder Hersteller haben wir nichts mehr zu tun.
Kaufen wir aber ein smartes Produkt, gehen wir mit dem Hersteller eine Art Ehe ein. Das Produkt ist auf Lebenszeit mit einem Background vernetzt, zum Beispiel mit einem Server auf einer Cloud-Plattform, der die Kühlschranktemperatur überwacht und steuert. Wir müssen darauf vertrauen, dass der Kühlschrank-Server des Herstellers während der kommenden zehn Jahre erreichbar ist. Doch wer kann uns das garantieren? Der Hersteller ist abhängig von Dritten, nicht zuletzt vom Betreiber der Cloud-Plattform. Und was passiert, wenn ein Glied dieser sensiblen Kette nach fünf Jahren in die Insolvenz geht? Damit unser Kühlschrank funktionsfähig und betriebssicher bleibt, sind wir auf Updates der Firmware angewiesen. Nicht mehr ganz taufrische Smartphones leiden schon heute darunter, dass das Betriebssystem nicht mehr unterstützt wird, die Hardware aber noch allen aktuellen Anforderungen genügt. Warum sollte das bei Kühlschränken anders sein?
Und was ist mit den Daten, die unser Kühlschrank sammelt? Erst kürzlich musste ein Hersteller von Smart-TVs eine hohe Geldbuße zahlen, weil er die Sehgewohnheiten von elf Millionen Käufern protokolliert und an Dritte veräußert hatte. Nicht weil die Daten gesammelt und versilbert wurden, sondern weil die Käufer nichts davon ahnten...

Streikende Traktoren

Beispiele dafür, dass Hersteller versuchen, Konsumenten über intelligente Geräte in den Griff zu bekommen, gibt es inzwischen viele. Der unmittelbare Verlust des Garadged-Käufers war die Zeit, die er mit dem vergeblichen Installieren vertan hat. Die mittelbaren Folgen können viel nachhaltiger sein.
Die Traktoren des amerikanischen Maschinenbauers John Deere sind längst nicht mehr simple landwirtschaftliche Fahrzeuge mit Verbrennungsmotor, sie sind ausgewachsene Computer auf Rädern. Beim Erwerb eines Traktors muss der Käufer Nutzungsbedingungen unterschreiben, die ihm strikt verbieten, an

seinem Eigentum zu schrauben. Befugt ist dazu nur das vom Hersteller zertifizierte Fachpersonal. Diese Techniker sind nicht nur immens teuer, die Werkstätten liegen meistens so weit entfernt, dass es lange dauern kann, bis ein Traktor repariert ist. Doch gerade während der Erntezeit zählt jede Minute, dann kosten streikende Traktoren besonders viel Geld.

Die Zeiten, als amerikanische Landwirte ihre Maschinen legal von örtlichen Kfz-Betrieben warten und reparieren ließen, gehören längst der Vergangenheit an. Der Journalist Jason Koebler sprach mit Landwirten, die sich an Hacker-Foren gewandt haben [1], um gekrackte Software für ihre Maschinen zu erstehen. Dort ist unter anderem ein Diagnoseprogramm downloadbar, das Probleme der Hardware oder Software lokalisiert. Das ist ein Tool, so meinten die Interviewten, das zum Lieferumfang jedes von John Deere verkauften Traktors gehören müsste.

Die gekrackte Software bringt den Landwirten zwar einen Teil der Autonomie über ihr Eigentum zurück, doch das hat seinen Preis. Die Betroffenen befürchten, dass sie von John Deere wegen Verletzung der Nutzungsbedingungen in Regress genommen werden. Das bedrohlichste Damoklesschwert ist jedoch, so vertrauten sie dem Journalisten Jason Koebler an, dass der Hersteller John Deere, agierend aus dem Hinterhalt, ihre Traktoren und Maschinen lahmlegen könnte.

An den potentiell streikenden Traktoren zeigt sich ebenso wie an der geschäftstüchtigen Tür aus dem Science-Fiction-Roman, dass smarte Geräte oder Maschinen tatsächlich intelligent sein können – entscheidend ist nur, zu wessen Nutzen! ◀

(160393)gd

Weblinks

[1] https://is.gd/mpeZUY

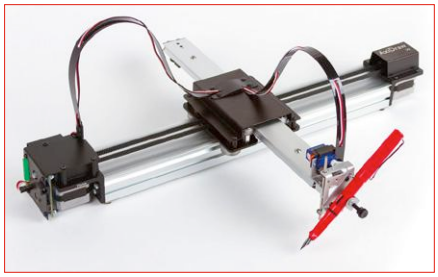
Puppenspieler (Foto: Public Domain).

Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

AxiDraw 3

AxiDraw ist ein moderner, schneller Plotter (XY: 28 cm/s) bzw. eine Art „Zeichenroboter", der auf jeder ebenen, zum Zeichnen geeigneten Fläche jedes beliebige Muster im skalierten Vektorgrafik-Format (.svg) anbringen kann – und dies sowohl vertikal als auch horizontal. Neben Papier zeichnet AxiDraw auch auf Karton, auf Tafeln, Platten, Schilder (auch vertikal), gerahmte Leinwand und vieles mehr. Bei Bedarf kann der Stift auch in einen Winkel von 45 ° gebracht werden, um zum Beispiel beim Schreiben mit einem Füller ein Schreiben per Hand nachzuahmen. Der Wechsel zu dieser Einstellung erfolgt manuell per Stellschraube. Das Produkt wurde vor etwa einem Jahr von Evil Mad Scientist auf den Markt gebracht, das seine Geräte in Kalifornien entwickelt und produziert, zum Beispiel den erstaunlichen EggBot zum Zeichnen auf ei- oder kugelförmigen Objekten und andere Geräte aus dem Grafikbereich.



Denis Meyer (Elektor-Labor)

www.elektor.de/axidraw-v3

Elektor-Bestseller

1. SmartScope
   www.elektor.de/smartscope

2. Andonstar USB-Mikroskop V160
   www.elektor.de/andonstar-v160
3. ESP8266-WLAN-Funkmodul
   www.elektor.de/150445-91
4. GSM/GPRS Projects
   www.elektor.de/gsm-gprs-projects
5. Raspberry Pi 3
   www.elektor.de/rpi-3
6. Serielle Schnittstellen
   www.elektor.de/serielle-schnittstellen
7. Arduino-Sanduhr
   www.elektor.de/sanduhr
8. StromPi 2
   www.elektor.de/strompi-2

Serielle Schnittstellen

Die serielle Schnittstelle dient dem Datenaustausch zwischen Mikrocontrollern und Peripheriegeräten. Das Wissen um diese seriellen Schnittstellen erweitert die Einsatzmöglichkeiten von Mikrocontrollern und Peripheriegeräten ungemein. In diesem Buch werden die Grundlagen der Kommunikation für die drei seriellen Schnittstellen I²C, SPI und 1-Wire vorgestellt.

Mitgliederpreis: 39,80 €

www.elektor.de/serielle-schnittstellen

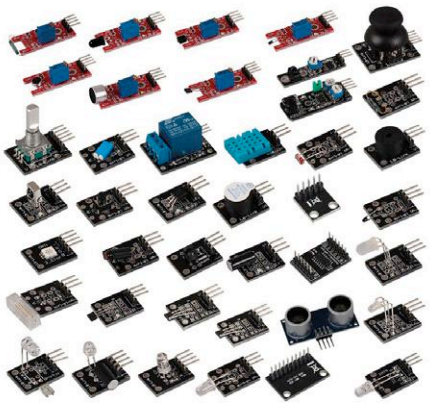
Das ESP8266-Praxisbuch

Der ESP8266 ist ein programmierbares WLAN-Funkmodul mit zahlreichen Schnittstellen wie UART, I²C und SPI. Das Board ist sehr preiswert und bereits für unter 3 Euro verfügbar. Die UART-Schnittstelle sorgt dabei für eine einfache Integration in Mikrocontrollerprojekte. Das ESP8266-Modul kann hervorragend mit dem Arduino zusammenarbeiten und ermöglicht ihm über die serielle Schnittstelle den Zugang zum Netzwerk und Internet.

Mitgliederpreis: 39,80 €

www.elektor.de/esp8266-buch

Sensor-Kit X40



Dieses hochwertige Sensor-Kit wurde speziell für die gängigsten Open-Source-Plattformen entwickelt. Es ist kompatibel zu Einplatinencomputern (Raspberry Pi, Banana Pi, CubieBoard, CubieTruck, Beaglebone, pcDuino) und Mikrocontrollern (Arduino, ATmega, AVR, PIC, STM32 usw.). Es beinhaltet insgesamt 40 verschiedene Sensoren. Sie können die Sensoren entweder fest verlöten oder auf ein Breadboard stecken.

Mitgliederpreis: 71,96 €

www.elektor.de/sensor-kit-x40

NEU!

Das MQTT-Praxisbuch

MQTT ist ein leichtgewichtiges, ereignis- und nachrichtenorientiertes Protokoll zur effizienten und asynchronen Kommunikation zwischen Geräten auch über limitierte Netzwerke. Das bereits 1999 von IBM entwickelte Protokoll eignet sich heute in besonderer Weise für IoT-Anwendungen. Im Gegensatz zu HTTP mit Request/Response-Verfahren ist bei MQTT eine Publish/Subscribe-Architektur umgesetzt. Es stehen mittlerweile zahlreiche MQTT-Broker und -Clients zur Verfügung. Aufgrund der Unterstützung durch die Eclipse Foundation, IBM und vieler anderer stehen zahlreiche Komponenten kostenlos im Internet zur Verfügung. Client-Bibliotheken gibt es für die unterschiedlichsten Plattformen und Programmiersprachen. Unterstützt werden u.a. die PC-Plattform mit Java und .Net sowie Arduino und Raspberry Pi.

Das MQTT-Praxisbuch führt Schritt für Schritt in die praktische Arbeit mit diesem ressourcensparenden Protokoll ein.

Mitgliederpreis: 29,80 € (frei Haus)
www.elektor.de/das-mqtt-praxisbuch

Digitale Systeme mit FPGAs entwickeln

FPGAs sind Standard-ICs, die dem Anwender die Konfiguration von Hardwarestrukturen ermöglichen. Mit FPGAs ist die Umsetzung digitaler Systeme ab Stückzahl eins machbar. Durch die Verfügbarkeit kostenloser Lizenzen und preiswerter Entwicklungsboards ist die finanzielle Einstiegsschwelle in diese Technik niedrig. Bei der Überwindung der fachlichen Schwelle hilft dieser 6-teilige Kurs.

Mitgliederpreis: 49,00 €

www.elektor.de/digitale-systeme-fpga

AVR-Programmierung für Quereinsteiger

Atmel AVR ist eine 8-bit-Mikrocontroller-Familie des Herstellers Atmel. Diese Controller sind wegen ihres einfachen Aufbaus, ihrer leichten Programmierbarkeit, den kostenlosen Entwicklungswerkzeugen und der Verfügbarkeit in DIP-Gehäuseformen auch bei Makern äußerst beliebt. Darüber hinaus sind diese Controller bereits ab zwei Euro erhältlich. Im Arduino Uno-Board wird der ATmega328 verwendet.

Mitgliederpreis: 34,80 €

www.elektor.de/avr-prog-fuer-quereinsteiger

StromPi 2

Die StromPi 2-Erweiterungsplatine ist die Neuentwicklung der beliebten Power-Solution StromPi 1. Nicht nur, dass der Spannungsbereich auf 6-61 V erhöht wurde, um z. B. größere/mehrere Batterien oder Spannungsversorgungen anzuschließen, so liefert StromPi 2 nun auch bis zu 3 A Strom, um auch größere Projekte mit genügend Strom beliefern zu können.

Mitgliederpreis: 26,96 €

www.elektor.de/strompi-2

Hexazener Sudoku für Elektroniker

Für diese Ausgabe haben wir uns ein ganz besonderes Hexadoku ausgedacht, in Form eines Zenerdioden-Symbols. Das entsprechende Gitter sieht etwas anders aus als sonst, wobei sich jedoch die Regeln für die erlaubten Zeichen und die Art, sie einzutragen, nicht geändert haben.
Zur Erinnerung: Ein Hexadoku-Rätsel setzt sich aus Hexadezimalzahlen zusammen (von 0 bis 9 und von A bis F). Jedes Zeichen darf dabei in jeder Reihe, jeder Spalte und in jedem Fach nur einmal erscheinen. Die Fächer sind durch eine fett gedruckte Umrandung gekennzeichnet.

Das Hexazener unterscheidet sich vom Hexadoku durch folgende Merkmale:

- Das Gitter enthält 16 Fächer, aber 18 Zeilen und 20 Spalten.
- Vier Zonen sind leer (weiß).
- Die Fächer sind nicht quadratisch oder rechteckig, sondern gleichen vielmehr einem Puzzle-Teil.
- Die Anzahl der Zellen eines Fachs hängt von dessen Form ab, überschreitet jedoch nicht die Zahl 16.
- Die Gesamtzahl der Felder beträgt 186, im Gegensatz zu 256 beim Hexadoku.

Da einige Fächer weniger als 16 Zellen besitzen, lassen sich dort nicht alle hexadezimalen Zahlen unterbringen - eine kleine, zusätzlich eingebaute Hürde auf dem Weg zur Lösung.
Die in den Feldern sichtbaren Ziffern legen die Anfangskonfiguration fest. Wenn Sie einen Gutschein gewinnen möchten, müssen Sie nicht das ganze Rätsel einsenden, sondern nur die 5 Ziffern in den grauen Feldern.

Einsenden Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

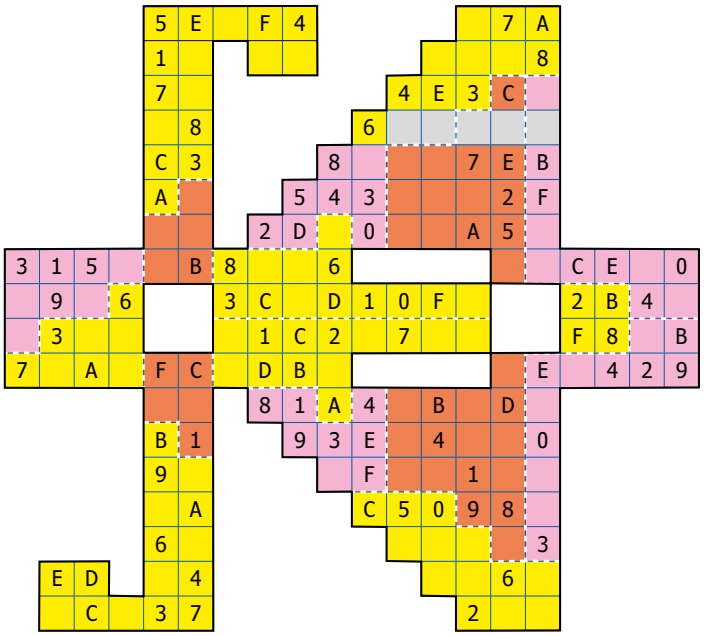
Einsendeschluss ist der 20. Juli 2017.

Die Gewinner des Hexadokus aus der Mai/Juni-Ausgabe stehen fest!

Die richtige Lösung ist: **1E709.**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen: Vincent Pierdominici, Adriaan Hoed, Nikos Chararas, Guy Savage und Jean Thevenoux.

Herzlichen Glückwunsch!



2	4	8	C	F	5	B	3	6	A	1	E	7	0	9	D
7	B	D	E	0	2	4	6	8	9	3	F	A	C	5	1
5	9	6	3	7	A	D	1	B	C	0	2	4	8	E	F
F	0	1	A	C	8	9	E	7	D	4	5	2	3	6	B
1	7	5	9	8	D	C	F	3	E	6	0	B	4	A	2
0	C	2	4	6	B	A	9	5	F	D	8	3	7	1	E
8	F	A	D	E	3	7	4	C	B	2	1	5	6	0	9
B	E	3	6	1	0	2	5	4	7	9	A	8	D	F	C
3	2	7	F	B	E	5	8	A	0	C	9	6	1	D	4
4	5	B	8	9	1	0	7	D	2	E	6	C	F	3	A
9	A	C	0	D	6	F	2	1	3	B	4	E	5	7	8
D	6	E	1	4	C	3	A	F	5	8	7	9	B	2	0
E	8	9	5	3	4	1	C	0	6	A	D	F	2	B	7
C	1	F	B	A	7	E	0	2	4	5	3	D	9	8	6
6	3	4	7	2	9	8	D	E	1	F	B	0	A	C	5
A	D	0	2	5	F	6	B	9	8	7	C	1	E	4	3

Der Rechtsweg ist ausgeschlossen. Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.

Join the elektor Community

Lesen Sie jetzt die europaweit umfassendste deutschsprachige Fachzeitschrift für Elektronik und Mikrocontroller in der ultimativen GOLD-Mitgliedschaft und profitieren Sie von allen Vorteilen!

Ihre Vorteile als GOLD-Mitglied:

- 6x Elektor-Doppelausgabe (132 Seiten!) pünktlich und zuverlässig frei Haus
- Jedes Heft gleichzeitig als PDF unterwegs lesen
- 6x Elektor Business gedruckt und digital
- Zugriff auf das gesamte Elektor-Archiv ab 1990
- Zugang zu Gerber-Dateien von über 1.000 Labs-Projekten
- Jahrgangs-DVD (Wert: 27,50 €) ohne Zuzahlung
- 10% Online-Rabatt auf alle Shop-Produkte

elektor GOLD Mitgliedschaft

elektor GREEN Mitgliedschaft

UMWELTSCHONEND – GÜNSTIG – GREEN

Möchten Sie Elektor lieber im elektronischen Format beziehen? Dann ist die GREEN-Mitgliedschaft ideal für Sie!

Die GREEN-Mitgliedschaft bietet (abgesehen von den Printausgaben) alle Leistungen und Vorteile der GOLD-Mitgliedschaft.

Jetzt Mitglied werden unter
www.elektor.de/mitglied

reichelt elektronik
Technik verbindet.

BESTELLT | BELIEFERT | BEGEISTERT

- ✓ Mehr als 75.000 Produkte
- ✓ Top-Preis-Leistungsverhältnis
- ✓ Hohe Verfügbarkeit und 24-h-Lieferservice
- ✓ Starke Marken und echte Qualität

reichelt.de | @reichelt_el | google.com/+reichelt

UNABHÄNGIGE STROMVERSORGUNG IDEAL FÜR UNTERWEGS

(Intenso)®

Intenso Slim-Powerbank, 10.000 mAh

Der Testsieger, bestens geeignet für Smartphones & Tablets!

- Zellenkapazität: 10.000 mAh, Lithium-Ionen
- Output-Anschluss: USB A und integriertes Micro-USB-Kabel
- Output: 5 V DC, max. 2,1 A
- Schutzeinrichtungen: Überspannungsschutz, Entladeschutz, Überladeschutz, Kurzschlusssicherung
- Maße: 130 x 70 x 16 mm

TESTSIEGER
Stiftung Warentest
test
SEHR GUT (1,5)
Im Test: 20 Powerbanks
Qualitätsurteil: 1 sehr gut, 10 gut, 6 befriedigend, 3 ausreichend
Ausgabe 6/2016
www.test.de
16GE95

statt 21,95
Bestell-Nr.: INTENSO 7332530
41% SPAREN
12,95

AUSWAHLHILFE

Smartphone
geeignet sind Powerbanks mit 1 - 1,5 A und 2.200 - 6.000 mAh

Tablet-PCs
geeignet sind Powerbanks mit mehr als 1,5 A und 6.000 - 20.000 mAh

Logilink Slim-Powerbank, 20.000 mAh

Der Energieriese unter den Powerbanks

- Ausgang: 5 V DC, max. 2,1 A
- LED-Ladestandanzeige (4 Segmente)
- Schutzfunktionen: Überladesicherung, Tiefenentladesicherung, Kurzschlussschutz
- 2 USB-Ladeanschlüsse
- Maße (LxBxH): 160 x 82 x 23 mm

DARF ES EIN WENIG MEHR SEIN?!

Bestell-Nr.: LOGILINK PA0086B
29,95

LogiLink®

Stabilisiertes Universalnetzteil, 12-24 V DC

goobay®

Versorgt Net- und Notebooks im Auto zuverlässig mit frischer Energie!

- Eingangsspannung: 12 - 24 V DC
- Ausgang: 15 - 20 V DC / max. 4730 mA
- inklusive 10 DC-Anschlußstecker für verschiedene Geräte
- Kurzschluss-, Überlast-, Überstrom- und Überhitzungsschutz
- integrierter USB-Ladeport, 5 V DC

Bestell-Nr.: NTS 90 USB
34,60

Über 97% zufriedene Kunden!
www.shopauskunft.de

Tagespreise · Preisstand: 29. 5. 2017, Preise in € inkl. gesetzl. MwSt., zzgl. Versandkosten, reichelt elektronik, Elektronikring 1, 26452 Sande (D)

JETZT NEWSLETTER ABONNIEREN & GEWINNEN!
Wir verlosen monatlich unter allen neuen Newsletter-Abonnenten ein technisches Highlight!
JETZT MITMACHEN ▶ http://rch.lt/v3

www.reichelt.de
BESTELLHOTLINE: +49 (0)4422 955-333